Variational Characterization of Shapley Values

David S. Rosenberg

Bloomberg ML EDU

December 8, 2021

Contents

Recap of Shapley values

Coalitional game[1]

- Suppose there is a game played by a team (or "coalition") of players.
- A **coalition game** is
  - a set N consisting of n "players" and
  - a function $v : 2^N \to \mathbb{R}$, with $v(\emptyset) = 0$, assigning a value to any subset of players.
- Suppose the whole team plays and gets value $v(N)$.
- How should that value be allocated to the individuals on the team?
- Is there a fair way to do it that reflects the contributions of each individual?

[1] Based on the Shapley value article in Wikipedia [Wik20] and [MP08].

- Where we're headed here is that we're going to apply this approach of "value allocation" to "coalitions" of feature "working together" to produce the final output.
- Of course, it's not really clear what it means to use a subset of features with a specific prediction function $f(x)$.
- Various approaches to this will give us different feature interpretations.

Solutions to coalition games

- Let $\mathcal{G}(N)$ denote the set of all coalition games on set N.
  - i.e. a game for every possible $v: 2^N \to \mathbb{R}$.
- A **solution** to the allocation problem on the set $\mathcal{G}(N)$ is a map $\Phi: \mathcal{G}(N) \to \mathbb{R}^n$
  - gives the allocation to each of n players for any game $v \in \mathcal{G}(N)$.
- The **Shapley value solution** is $\Phi(v) = (\phi_i(v))_{i=1}^n$ where

$$\phi_i(v) = \sum_{S \subset (N - \{i\})} k_{|S|,n} \left(v(S \cup \{i\}) - v(S)\right),$$

where $k_{s,n} = s!\,(n-s-1)!/n!$.

The Shapley value solution is special

The Shapley value solution is the unique solution with the following properties:

- **Efficiency**: For any $v \in \mathcal{G}(N)$, $\sum_{i \in N} \phi_i(v) = v(N)$.
- **Symmetry**: For any $v \in \mathcal{G}(N)$, $\forall i, j \in N$, if $v(S \cup \{i\}) = v(S \cup \{j\})$ for every subset S of players that excludes i and j, then $\phi_i(v) = \phi_j(v)$.
- **Linearity**: For any $v, w \in \mathcal{G}(N)$, we have $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$ for every player i in N. Also, for any $a \in \mathbb{R}$, $\phi_i(av) = a\phi_i(v)$ for every player i in N.
- **Null**: A player i is **null** in v if $v(S \cup \{i\}) = v(S)$ for all coalitions $S \subset N$. If player i is null in a game v, then $\phi_i(v) = 0$.
- That's all very nice... but doesn't give much intuition on what the values are.
- Shapley values are the sum of **exponentially many terms, with mysterious weights**.

Reformulation of Shapley values

Approximating a set function

- Suppose we have an set function $v(S)$
  - acting on sets $S \subset N = \{1, \ldots, n\}$.
- We assume it's arbitrary, except that $v(\emptyset) = 0$.
- So there are $2^n - 1$ degrees of freedom.
- The set function has all the information we can want about
  - how valuable each player is in the context of other player coalitions.
- But it's too complicated to examine directly.
- What if we approximate $v(S)$ by a simpler set function?

Additive set function

- One simple type of set function is an **additive set function.**
- On a finite set, an additive set function takes the form

$$w(S) := \sum_{i \in S} w_i,$$

for some $w \in \mathbb{R}^n$. (empty sum is 0).

Notation overload

Note that when we write $w(S)$, we're thinking of w as a function. While a plain w (without parenthesis), we're thinking of $w \in \mathbb{R}^n$. The relation between the two is in the definition of $w(S)$ above. So for any $w \in \mathbb{R}^n$, we get a function $w : S \mapsto \mathbb{R}$.

Coalitional games given by additive set functions

- On a finite set, an additive set function takes the form

$$w(S) := \sum_{i \in S} w_i,$$

  for some $w \in \mathbb{R}^n$. (empty sum is 0).
- Each element of the set $N = \{1, \ldots, n\}$ gets a value,
  - and $w(S)$ is just the sum of the values in the set S.
- A coalitional game represented by $w(S)$ is easy to interpret:
  - A reasonable assessment of the value of player i is w_i.
- If $w(S) \approx v(S)$,
  - perhaps we can apply interpretations of $w(S)$ to $v(S)$.

Fitting a set function

- We want to find an additive $w(S)$ that approximates $v(S)$.
- We can find $w \in \mathbb{R}^n$ solving

$$\operatorname*{arg\,min}_{w \in \mathbb{R}^n} \sum_{S \subset N} [w(S) - v(S)]^2 \, q(|S|),$$

  where $q : \{1, \ldots, n-1\} \to \mathbb{R}$ is an arbitrary **weight function**.
- This is a weighted least squares fit of $w(S)$ to $v(S)$.
- Note that the weight corresponding to S depends only on the size of S.
  - e.g. maybe we could weight large sets more than small sets
- Amazingly, we can derive a closed form solution to this optimization problem,
  - with the additional constraint that $w(N) = v(N)$...

We've intentionally left $q(0)$ and $q(n)$ undefined, as they won't matter...

Generalized Shapley values

Theorem ([CGKR88, Thm 3])

The $w \in \mathbb{R}^n$ that minimizes

$$J(w) \;=\; \sum_{S \subset N} \left[w(S) - v(S)\right]^2 q(|S|),$$

subject to the constraint that $w(N) = v(N)$ is

$$w_i = \frac{v(N)}{n} + \frac{1}{\beta}\left(\sum_{S \subset N : i \in S} v(S) q(|S|) - \frac{1}{n}\sum_{j=1}^{n} \sum_{S \subset N : j \in S} v(S) q(|S|)\right),$$

where $\beta = \sum_{s=1}^{n-1} q(s)\binom{n-2}{s-1}$, provided $\beta \neq 0$.

- The w_i's are called **generalized Shapley values**.

- Note that $q(0)$ and $q(n)$ can take any values in the objective function without affecting the results since
  - $w(\emptyset) = v(\emptyset) = 0$ by definition of w and v, and
  - $w(N) = v(N)$ by the constraint.

A quadratic optimization for Shapley values

- Suppose we choose $q(s) = c\binom{n-2}{s-1}^{-1}$, for any $c \neq 0$.
  - We'll call this the **Shapley weight function**.
- Then the $w \in \mathbb{R}^n$ that minimizes
  - the quadratic objective $J(w) = \sum_{S \subset N} [w(S) - v(S)]^2 q(|S|)$,
  - subject to the constraint $w(N) = v(N)$
- are **exactly the Shapley values**! [CGKR88, Thm 4].

Takeaway

Shapley values arise from finding the best possible additive approximation to a given set function, for a very particular definition of "best possible."

- Practical issue: The objective function has 2^n terms.

Theorem ([CGKR88, Thm 4])

If we choose $q(s) = c\binom{n-2}{s-1}^{-1}$ for any $c \neq 0$, then the solution to the constrained optimization problem stated above is

$$\begin{aligned} w_i &= \frac{1}{n} \sum_{S \subset N : i \in S} \binom{n-1}{|S|-1}^{-1} [v(S) - v(S - \{i\})] \\ &= \sum_{S \subset (N - \{i\})} k_{|S|,n} [v(S \cup \{i\}) - v(S)], \end{aligned}$$

where $k_{s,n} = \frac{1}{s+1}\binom{n}{s+1}^{-1} = \frac{1}{n}\binom{n-1}{s-1}^{-1} = s!(n-s-1)!/n!$. And so w_i are the Shapley values.

- The notation in [CGKR88, Thm 4] is different, but here we've translated it to our notation. There are many equivalent formulations of Shapley values, so we've tried to give a variety here. The last one matches our original definition.
- With a few lines of algebra and taking $c = \frac{1}{n}$, one can show this result is equivalent to [LL17, Thm 2].

Finding the Shapley values

Dropping the constraint

- The generalized Shapley value objective is

$$J(w) = \sum_{S \subset N} [w(S) - v(S)]^2 q(|S|).$$

- We have the constraint that $w(N) = \sum_{i=1}^{n} w_i = v(N)$.
- An easy way to enforce the equality constraint is to eliminate a variable.
- Let's eliminate a variable by setting $w_n = v(N) - \sum_{i=1}^{n-1} w_i$.
- With this substitution, we no longer need an explicit constraint that $w(N) = v(N)$.
- We can take $q(0) = q(n) = 0$, without affecting the result.

The weights for the Shapley kernel

- For $n = 30$, let $q(s) = \binom{n-2}{s-1}^{-1}$ be the Shapley weight function.

/Users/drosen/Dropbox/repos/mltopics/code/interpretation/figures/shapley_weigh

- This is a plot of the weight function $q(s)$ on $\{1, 2, \ldots, 29\}$. As noted, we can take $q(0) = q(30) = 0$ in our optimization.
- So clearly the largest and smallest sets have enormously more weight than mid-sized sets.
- But also remember that there are enormously more mid-sized sets than very small and very large sets...

Total weight by subset size

- For $n = 30$, let $w(s) = \binom{n}{s} q(s)$ be the total weight for subsets of each size.

/Users/drosen/Dropbox/repos/mltopics/code/interpretation/figures/tot_weight_pe

- This is a plot of the total weight for all subsets of each size s, on $\{1, 2, \ldots, 29\}$.
- That is, we're plotting $w(s) = \binom{n}{s} q(s) = \binom{n}{s} \binom{n-2}{s-1}^{-1}$

$$
\begin{aligned}
w(s) = \binom{n}{s} \binom{n-2}{s-1}^{-1} &= \frac{n!}{s!(n-s)!} \frac{(s-1)!(n-s-1)!}{(n-2)!} \\
&= \frac{n(n-1)}{s(n-s)}
\end{aligned}
$$

- So most of the weight is on the small and large sets, but there is nontrivial weight throughout the range.

Monte Carlo approximation

- The objective function has 2^n terms,
  - which quickly becomes too large to handle exactly.

$$J(w) \quad = \quad \sum_{S \subset N} [w(S) - v(S)]^2 \, q(|S|).$$

- Since $q(|S|) > 0$ for the Shapley weight function,
  - we can normalize it into a distribution on subsets of N.
- Define $p(S) := \frac{1}{k} q(|S|)$. Then

$$J(w) \propto \mathbb{E}_{S \sim p(S)} [w(S) - v(S)]^2 .$$

- (How can you sample from $p(S)$? See note...)
- So now we can approximate $J(w)$ using as many subsets as we have patience.

- Rather than sampling from $p(S)$ directly, it's easier to first sample the size of S, and then we just sample uniformly from subsets of the selected size.
- Computing the probability distribution over sizes is straightforward by renormalizing $w(s) = \binom{n}{s} q(s)$ over $s \in \{1, \ldots, n-1\}$.

Hybrid approximation

- In the next module we'll discuss Kernel SHAP.
- Kernel SHAP computes Shapley values with a variant of the Monte Carlo approach.
- We can break the computation $J(w)$ into pieces, such as

$$J_2(w) := \sum_{S:|S| \in \{1,2,(n-2),n-1)\}} [w(S) - v(S)]^2 q(|S|).$$

  and

$$J_{-2}(w) := \sum_{S:3 \leqslant |S| \leqslant (n-3)} [w(S) - v(S)]^2 q(|S|),$$

  where $J(w) = J_2(w) + J_{-2}(w)$.
- We can then compute $J_2(w)$ by direct computation, and estimate $J_{-2}(w)$ by Monte Carlo.
- Of course we can change the the subset sizes we're using in direct computation.

- From the code for Kernel SHAP (December 8, 2021), it seems that they have a budget (can be user-provided) for the number of subsets they're going to sum over. They start with subsets of size 1 and $N-1$, and see if they can fit **all** of those subsets into their budget. If so, they sum over all those subsets explicitly.
- Next, they check if they can also include all subsets of size 2 and $N-2$ within the remaining budget. If so, they add those in. They continue until they get to an i for which they cannot fit all the subsets of size i and $N-i$ within the remaining budget. At this point, they switch to random sampling from remaining subsets with the remaining budget.

References

Resources

- The ideas in the reformulation of Shapley values and generalized Shapley values are from [CGKR88]. However, I found Yuchen Pei's blog post to be very helpful in understanding the proofs in the paper by Charnes et al [CGKR88].

References I

[CGKR88] A. Charnes, B. Golany, M. Keane, and J. Rousseau, *Extremal principle solutions of games in characteristic function form: Core, chebychev and shapley value generalizations*, pp. 123–133, Springer Netherlands, Dordrecht, 1988.

[LL17] Scott Lundberg and Su-In Lee, *A unified approach to interpreting model predictions*, 2017, pp. 4765–4774.

[MP08] Stefano Moretti and Fioravante Patrone, *Transversality of the shapley value*, TOP **16** (2008), no. 1, 1–41.

[Wik20] Wikipedia contributors, *Shapley value — Wikipedia, the free encyclopedia*, 2020, [https://en.wikipedia.org/wiki/Shapley_value; accessed 26-April-2021].