Lecture 19: Explainability: SHAP

Trustworthy Machine Learning

Spring 2024

Explainability

- Recap:
  - Introduction to explainable ML
  - Feature attribution problem
  - LIME (Local Interpretable Model-agnostic Explanations) algorithm
- Today: SHAP methods based on cooperative game theory
- Coming up:
  - Saliency maps
  - Formal guarantees for feature attribution methods
  - Counterfactuals
  - Rule synthesis
  - Data attribution methods: Influence functions, Datamodels

Today's Agenda

- Feature Attribution Problem: Given an input x and model f, find a subset of (interpretable) features of x that contribute the most to prediction f(x)

- Today: Explanation method SHAP
  - Cooperative game theory and Shapley values
  - Application to feature attribution problem
  - Efficient algorithm to approximate computation

- Resources:
  - Talk slides by Su-In Lee (U. Washington)
  - A unified approach to interpreting model predictions

    Lundberg and Lee; NeurIPS 2017

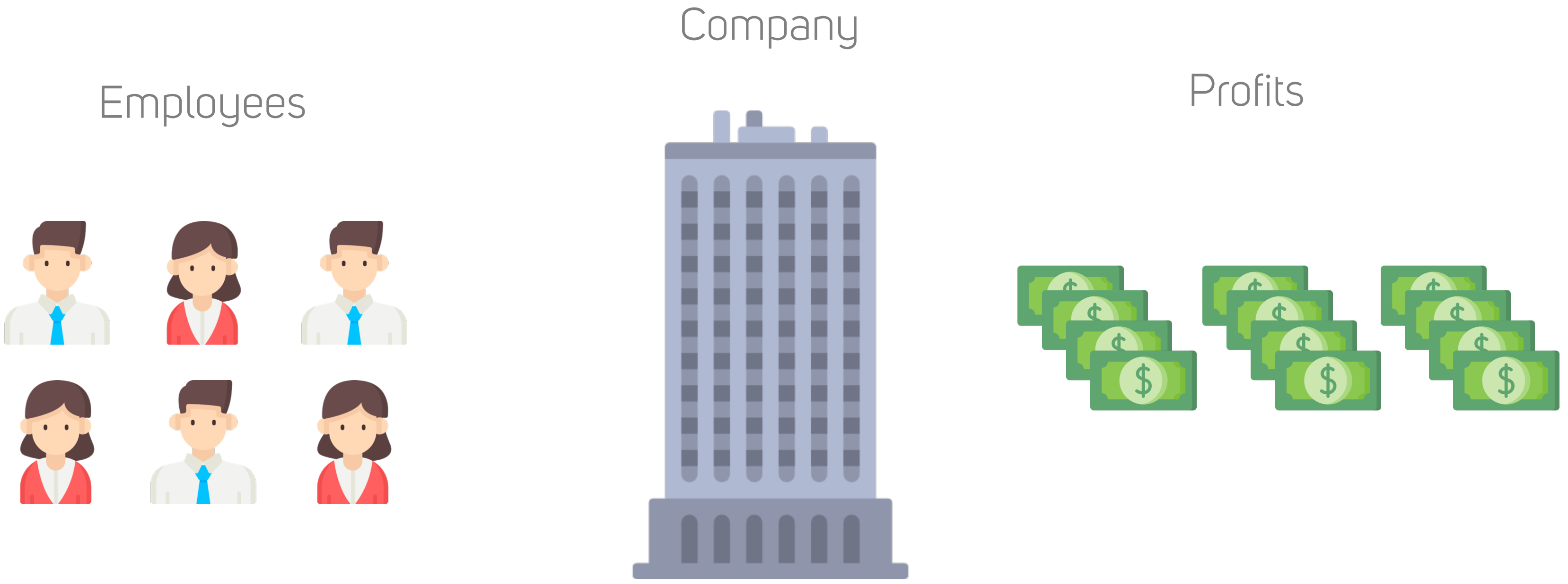
Cooperative game notation

- Set of *players* $D = \{1, \ldots, d\}$
- A *game* is given by specifying a value for every coalition $S \subseteq D$
- Mathematically represented by a *characteristic function*:

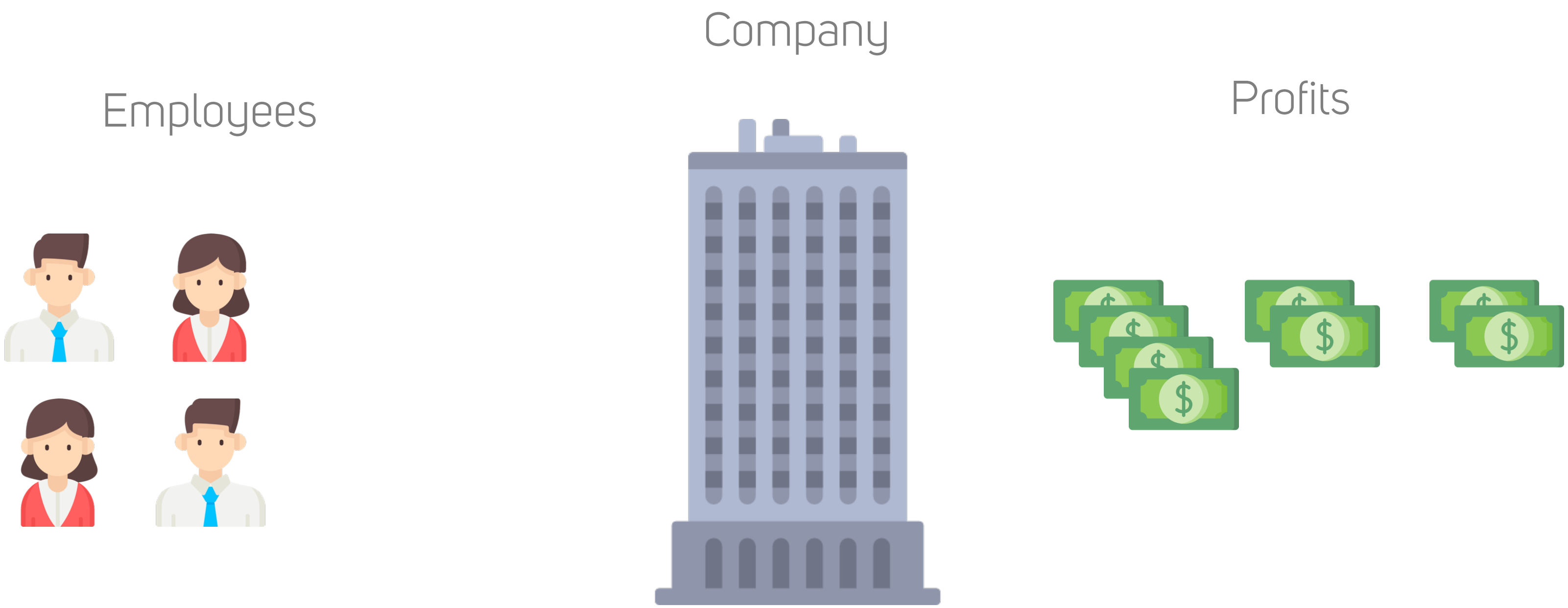
$$v: 2^D \mapsto \mathbb{R}$$

- Grand coalition value $v(D)$, null coalition $v(\emptyset)$, arbitrary coalition $v(S)$

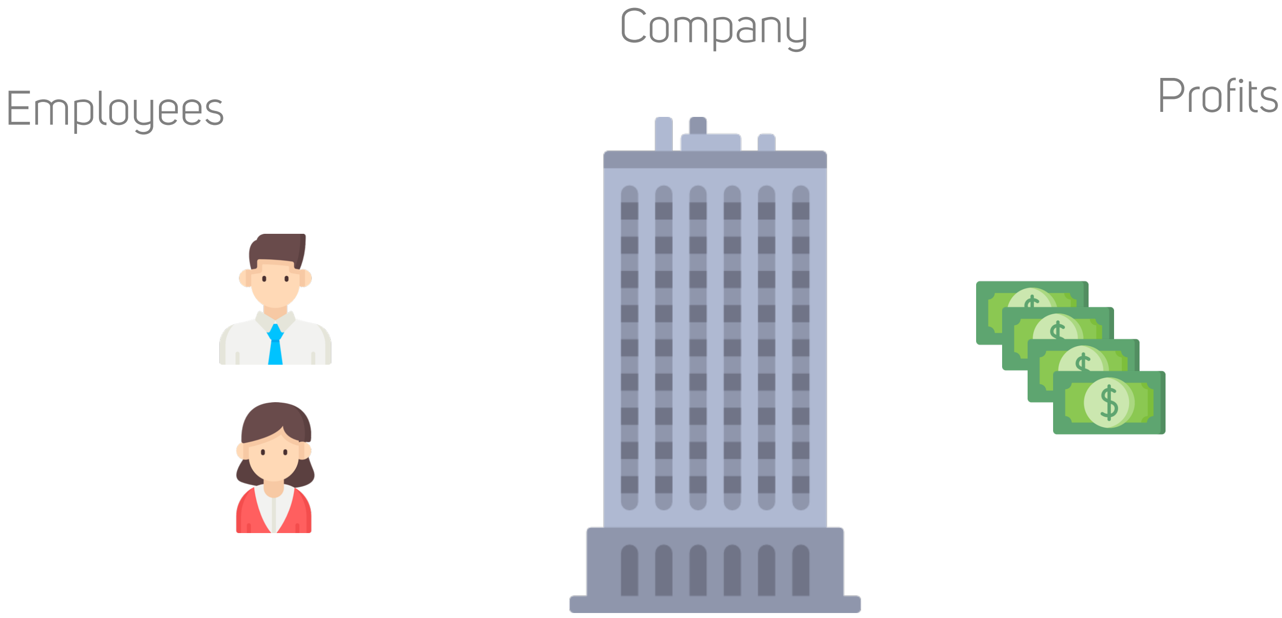
Company example



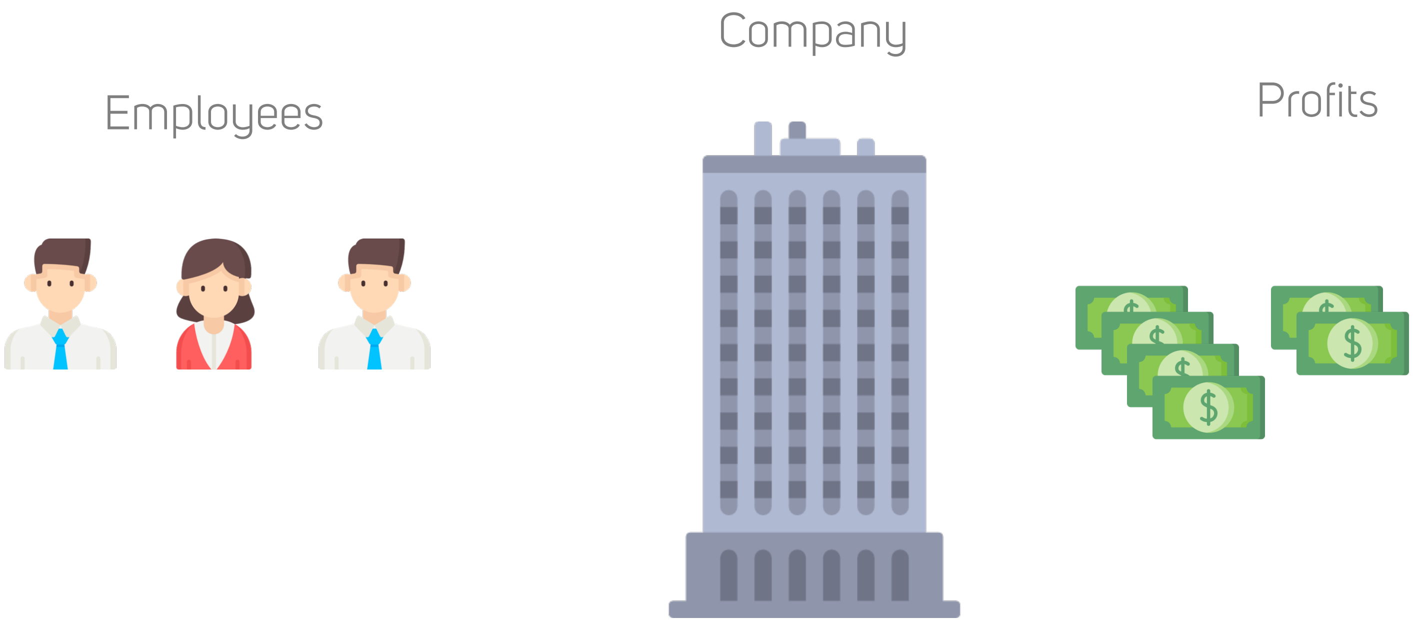
Company example



Company example



Company example

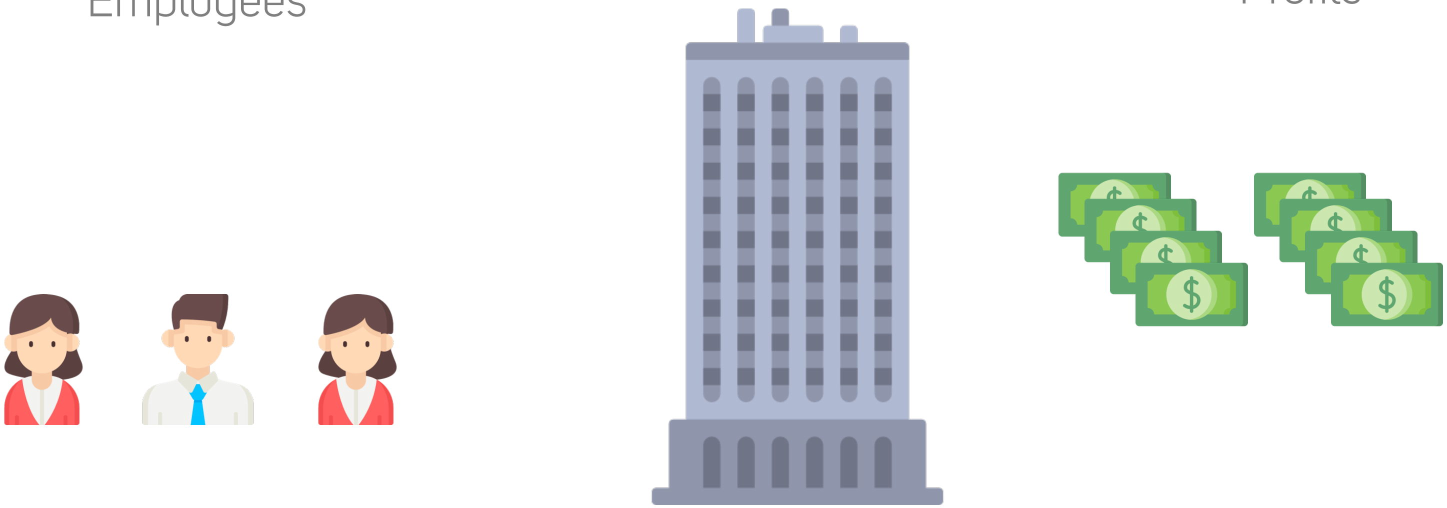


Company example

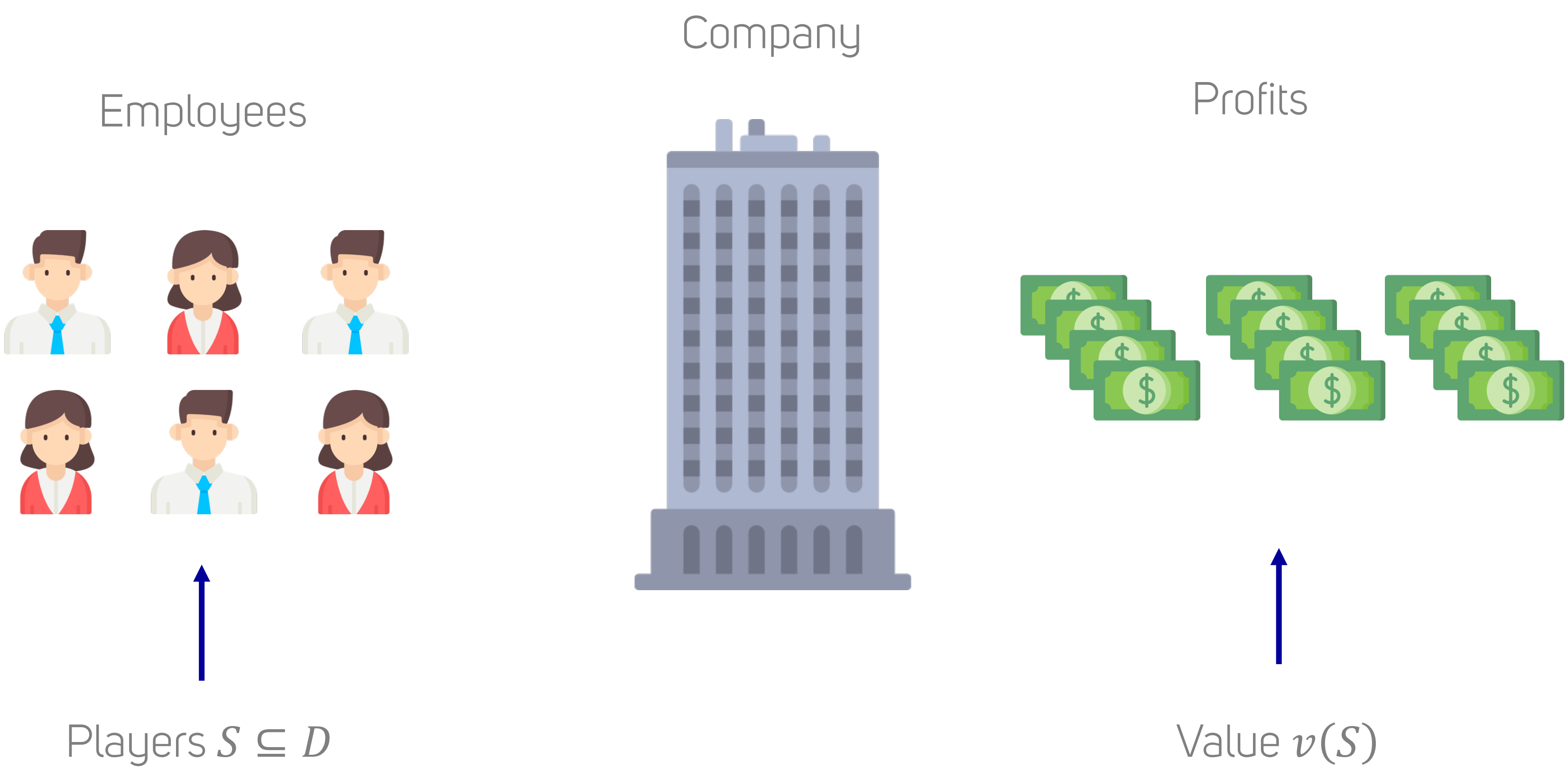
Company

Employees

Profits



Company example



Key game theory questions

- Which players will participate vs. break off on their own?
- How to allocate credit among players?

Shapley value

- A technique for allocating credit to players in a cooperative game
- Famously derived from a set of *fairness axioms*

Lloyd Shapley

- Won 2012 Nobel Memorial Prize in economics

Shapley value setup

- Let G denote the set of games on d players
- The Shapley value assigns a vector of credits to each game (in $\mathbb{R}^d$, one credit per player)
- Mathematically, a function of the form

$$\phi: G \mapsto \mathbb{R}^d$$

- For a game v, Shapley values are $\phi_1(v), \ldots, \phi_d(v)$

Shapley Value Example

- Players: owner o and n symmetric employees

- Coalition values:

  v(S) = 0 if S doesn't include owner o and = (|S|-1)p if S includes owner o

- Grand coalition value: np

- Game theory question: How should profit np be shared ?

Shapley Value Example

- Players: owner o and n symmetric employees
- Coalition values:

  v(S) = 0 if S doesn't include owner o and = (|S|-1)p if S includes owner o
- Grand coalition value: np
- Game theory question: How should profit np be shared ?

- Answer: Shapley values of each player give contribution of that player to total profit
- Owner's value = np/2, each employee's value = p/2

Another Example

- Players: owner o and n symmetric employees

- Coalition values:

  v(S) = 0 if S doesn't include owner o and at least one employee,

  = p otherwise (i.e. the owner and at least one employee shows up)

- Grand coalition value: p

- Game theory question: How should profit p be shared ?

Fairness axioms

Consider a game v and credit allocations $\phi(v) = [\phi_1(v), \dots, \phi_d(v)]$. We want to satisfy the following properties:

- *(Efficiency)* The credits sum to the grand coalition's value, or $\sum_{i \in D} \phi_i(v) = v(D) - v(\emptyset)$
- *(Symmetry)* If two players (i, j) are interchangeable, or $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq D$, then $\phi_i(v) = \phi_j(v)$
- *(Null player)* If a player contributes no value, or $v(S \cup \{i\}) = v(S)$ for all $S \subseteq D$, then $\phi_i(v) = 0$
- *(Linearity)* The credits for linear combinations of games behave linearly, or $\phi(c_1 v_1 + c_2 v_2) = c_1 \phi(v_1) + c_2 \phi(v_2)$, where $c_1, c_2 \in \mathbb{R}$

Lloyd Shapley, "A value for n-person games" (1953)

Axiomatic uniqueness

- The Shapley value (SV) is the only function $\phi: G \mapsto \mathbb{R}^d$ to satisfy these properties
- Given by the following equation:

$$\phi_i(v) = \sum_{S \subseteq D \setminus \mathrm{i}} \frac{|S|!\,(d - 1 - |S|)!}{d!} [v(S \cup \{i\}) - v(S)]$$

Weighted average across $S \subseteq D \setminus i$

Contribution from adding player i

Interpretation

- Intuitive meaning in terms of player orderings
  - Given an ordering π, each player contributes when added to the preceding ones
  - SV is the average contribution across all orderings

$$\phi_i(v) = \frac{1}{d!} \sum_{\pi \in \Pi} [v(\{ j \mid \pi^{-1}(j) \leq \pi^{-1}(i) \}) - v(\{ j \mid \pi^{-1}(j) < \pi^{-1}(i) \})]$$

Average across all orderings

Players up to and including i

Players preceding i

Example Shapley Value Calculation

- Players: owner o and n symmetric employees
- Coalition values:

  v(S) = 0 if S doesn't include owner o and at least one employee,

  = p otherwise (i.e. the owner and at least one employee shows up)

- Number of permutations = (n+1)!
- Permutations where owner's marginal contribution is 0
- Permutations where owner's marginal contribution is p = (n+1)! – n!
- Owner's Shapley value = [(n+1)!-n!]p/(n+1)! = [n/(n+1)] p
- Each employee's Shapley value = [1/n(n+1)] p

Application to ML

- Consider **features** as **players**
- Consider **model behavior** as **profit**
  - E.g., the prediction, the loss, etc.
- Then, use Shapley values to quantify each feature's impact

SHAP

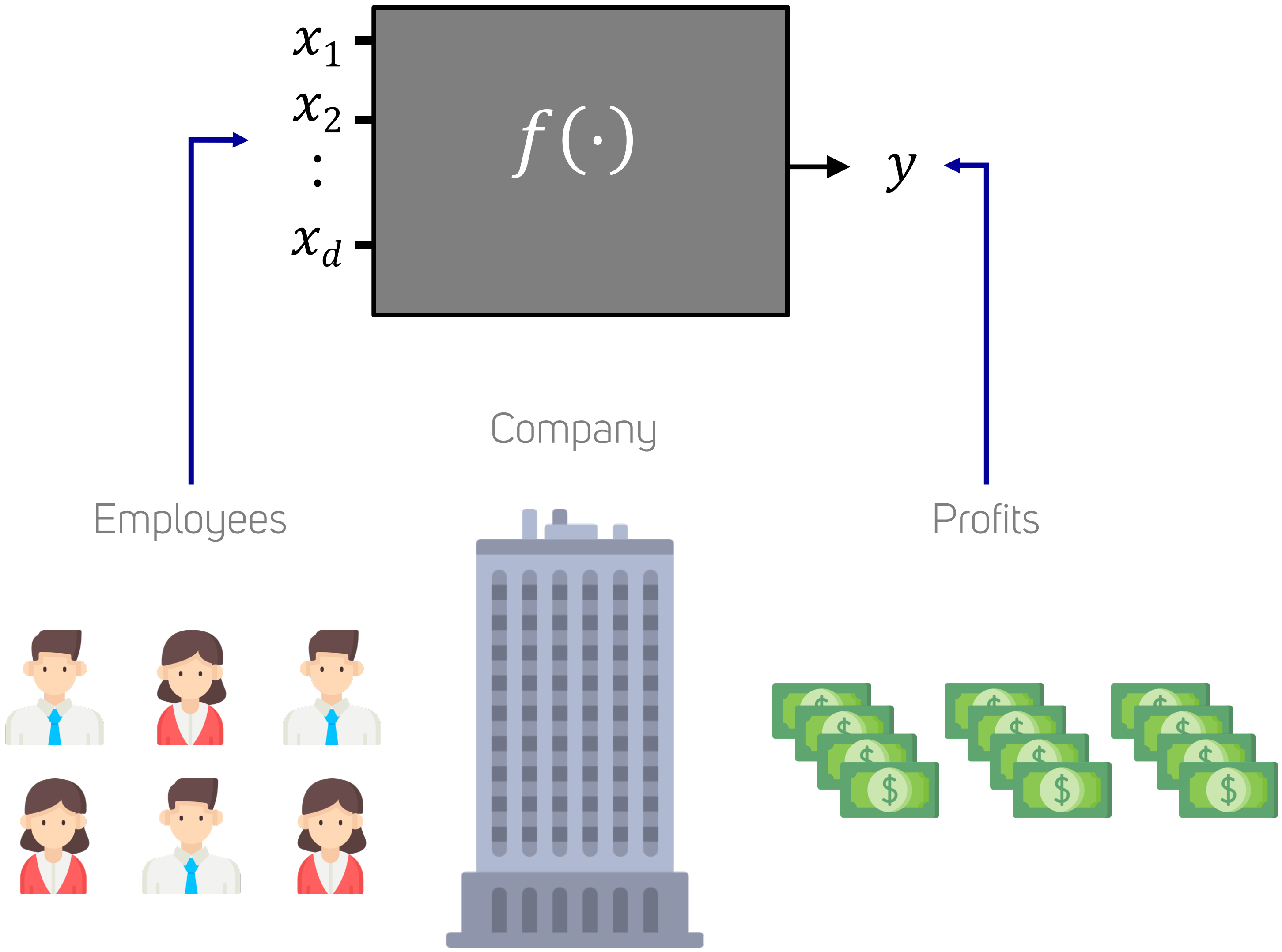
- SHAP = *SHapley Additive exPlanations*

- Popularized use of Shapley values in ML
  - Also used in earlier work by Lipovetsky & Conklin (2001), Strumbelj et al. (2009), Datta et al. (2016)

- SHAP uses Shapley values to explain individual predictions

Lundberg & Lee, “A unified approach to interpreting model predictions” (2017)

ML model



SHAP as a removal-based explanation

Recall the three choices for removal-based explanations:

1. **Feature removal:** $F(x_S) = \mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})]$
2. **Model behavior:** $v(S) = F_y(x_S)$
3. **Summary:** $a_i = \phi_i(v)$

Consider this more closely in the next slide

Shapley value

Notation clarification

- What is $\mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})]$?
- The expected value of the model output when conditioned on the feature values x_S

$$
\begin{aligned}
F(x_S) &= \mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})] \\
&= \mathbb{E}[f(x_S, x_{\bar{S}}) \mid x_s] \\
&= \sum_{x_{\bar{S}}} f(x_S, x_{\bar{S}}) \cdot \mathrm{p}(x_{\bar{S}} \mid x_S)
\end{aligned}
$$

Summation over all possible $x_{\bar{S}}$ values

Model output given $x_{\bar{S}}$

Probability of $x_{\bar{S}}$ conditioned on x_S

Notation clarification (cont.)

- Recall Bayes rule for conditional probability:

$$\mathrm{p}(x_{\bar{S}} \mid x_S) = \frac{\mathrm{p}(x_S, x_{\bar{S}})}{\mathrm{p}(x_S)}$$

Probability of $x_{\bar{S}}$ and x_S occurring together

Probability of x_S occurring on its own

Notation clarification (cont.)

- **Intuition:** in SHAP, we want to evaluate the model given a subset of features as follows
  - Fix the example to be explained x and the set of available features x_S
  - Withhold the remaining feature values $x_{\bar{S}}$
  - To do so, consider *all possible values* for $x_{\bar{S}}$, and make the corresponding predictions $f(x_S, x_{\bar{S}})$
  - Then average these predictions, weighting them according to the conditional probability $\mathrm{p}(x_{\bar{S}} \mid x_S)$

SHAP summary

- SHAP analyzes individual predictions by setting up the following cooperative game:

$$v(S) = F_y(x_S) = \mathbb{E}_{x_{\bar{S}} | x_S}[f(x_S, x_{\bar{S}})]$$

- Then determines feature attributions using the Shapley value:

$$a_i = \phi_i(v)$$

Practical alternative

- The conditional distribution is hard to estimate
- Instead, we can marginalize out features using their **marginal distribution**

$$\mathbb{E}_{x_{\bar{S}} | x_S}[f(x_S, x_{\bar{S}})] \approx \mathbb{E}_{x_{\bar{S}}}[f(x_S, x_{\bar{S}})]$$

↑ Drop conditioning

Remark

- In general, the conditional and marginal distributions are not equal

$$p(x_{\bar{S}} \mid x_S) \neq p(x_{\bar{S}})$$

- Assuming they're identical = assuming feature independence
- Can result in unlikely, *off-manifold* feature combinations

Marginal distribution

- Easy to implement with Monte Carlo estimation
- Choose m datapoints $x^1, \dots, x^m$ from dataset
- Approximate as follows:

$$\mathbb{E}_{x_{\bar{S}}}[f(x_S, x_{\bar{S}})] = \sum_{x_{\bar{S}}} \mathrm{p}(x_{\bar{S}}) f(x_S, x_{\bar{S}}) \approx \frac{1}{m} \sum_{i=1}^{m} f(x_S, x_{\bar{S}}^i)$$

Remark: permutation tests do this, but using a single sample

Setup

- Assume we have a game $v: 2^D \mapsto \mathbb{R}$
- We want to calculate Shapley values
- How straightforward is this?

Computational complexity

- The equation for Shapley values is:

$$\phi_i(v) = \sum_{S \subseteq D \setminus \mathrm{i}} \frac{|S|!\,(d-1-|S|)!}{d!}[v(S \cup i) - v(S)]$$

Summation across 2^{d-1} subsets

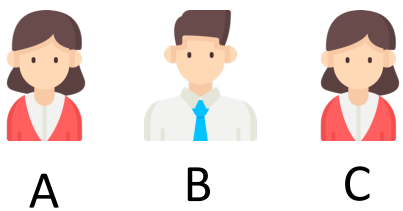
- Exponential running time $\mathcal{O}(2^d)$
- Intractable for even moderate d (e.g., $d > 20$)

What can we do?

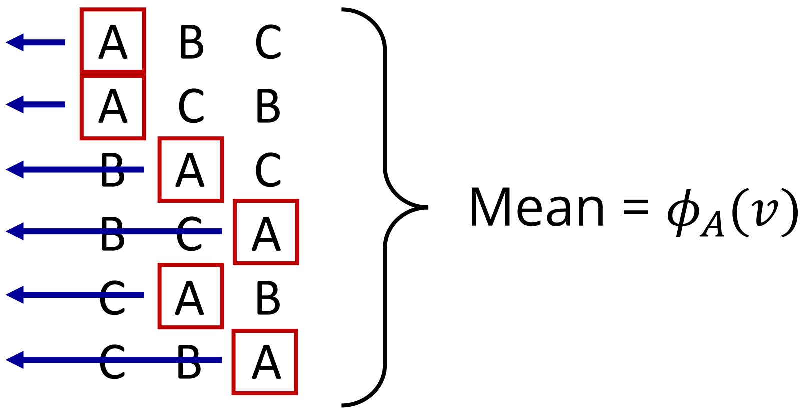
- We cannot calculate Shapley values exactly when d is large
- Instead, we can approximate them
- We'll discuss the following approaches:
  - Permutation-based estimation
  - Regression-based estimation
  - Others (briefly)

Permutation view

- Recall the Shapley value's ordering interpretation
- The value $\phi_i(v)$ is player i's average contribution across all player orderings



1. Enumerate all orderings
2. Find player contribution
3. Average



Permutation-based estimation

- **Problem:** $d!$ orderings is too many for large values of d

- **Idea:** sample a moderate number of orderings
  - Calculate average contributions across those orderings

Permutation-based estimation (cont.)

Algorithm 1: Permutation estimation

Input: Game v, iterations $m > 0$
Output: Shapley value estimates $\hat{\phi}_1(v), \ldots, \hat{\phi}_d(v)$
initialize $\hat{\phi}_i(v) = 0$ for $i = 1, \ldots, d$
for $j = 1$ **to** m **do**
    sample permutation $\pi \in \Pi$ uniformly at random
    $S = \varnothing$
    $\texttt{prev} = v(\varnothing)$
    for $k = 1$ **to** d **do**
        $i = \pi(k)$ `// Get next player in ordering`
        $S = S \cup \{i\}$
        $\texttt{curr} = v(S)$
        $\hat{\phi}_i(v) = \hat{\phi}_i(v) + \Big(\texttt{curr} - \texttt{prev}\Big)$ `// Update estimate`
        $\texttt{prev} = \texttt{curr}$
    end
end
set $\hat{\phi}_i(v) = \frac{\hat{\phi}_i(v)}{m}$ for $i = 1, \ldots, d$ `// Normalize`
return $\hat{\phi}_1(v), \ldots, \hat{\phi}_d(v)$

Regression view

- An alternative Shapley value characterization
- Perhaps surprisingly, SVs are the solution to a weighted least squares problem

Regression view (cont.)

- Consider a game $v: 2^D \mapsto \mathbb{R}$
- Consider a weighting function $\mu(S)$:

$$\mu(S) = \frac{d-1}{\binom{d}{|S|}\,|S|(d-|S|)}$$

- Shapley values minimize the following objective:

$$\min_{\beta_0,\ldots\beta_d} \sum_{S \subseteq D} \mu(S) \left(\beta_0 + \sum_{i \in S} \beta_i - v(S) \right)^2$$

Squared error

Weighted summation

Additive approximation

Regression-based estimation

- **Problem:** WLS problems are easy to solve, but 2^d terms is too many

- **Idea:** approximate WLS problem by sampling subsets according to $\mu(S)$
  - Incorporate weights $\mu(\emptyset) = \mu(D) = \infty$ as constraints, $\beta_0 = v(\emptyset)$ and $\sum_{i \in D} \beta_i = v(D) - v(\emptyset)$
  - Solve the constrained least squares problem

Regression-based estimation (cont.)

- Omitting a detailed algorithm here
  - Constraints make things a bit complicated
  - Method known as **KernelSHAP**, introduced by Lundberg & Lee (2017)
  - See paper below for relatively simple exposition

Covert & Lee, "Improving KernelSHAP: Practical Shapley value estimation via linear regression" (2021)

Conclusion

- Shapley values are an elegant idea from game theory
- Now used by multiple XAI methods, most famously by SHAP for individual predictions
- Leads to computational challenges, so we use approximations in practice
  - Simulate feature removal
  - Approximate Shapley values

Explainability

- Last lecture:
  - Introduction to explainable ML
  - Feature attribution problem
  - LIME (Local Interpretable Model-agnostic Explanations) algorithm
- Today: SHAP methods based on cooperative game theory
- Coming up:
  - Saliency maps
  - Formal guarantees for feature attribution methods
  - Counterfactuals
  - Rule synthesis
  - Data attribution methods: Influence functions, Datamodels