Beyond axis-aligned splits: A new BART prior for structured categorical predictors

Sameer K. Deshpande

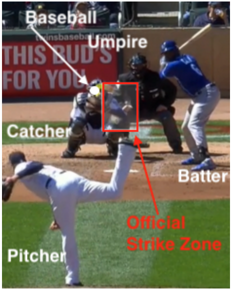
University of Wisconsin–Madison

ISBA 2022

28 June 2022

Problem 1: Pitch framing in baseball

- Umpires' ball/strike decisions not deterministic
- Some players can influence umpires' decisions
- Some catchers can increase $\mathbb{P}(\texttt{strike})$ & "turn balls into strikes"
- $\mathbb{P}(\texttt{strike})$ depends on complex interactions b/w pitch location, players, and umpires
- Let's use BART to flexibly fit $\mathbb{P}(\texttt{strike})$!

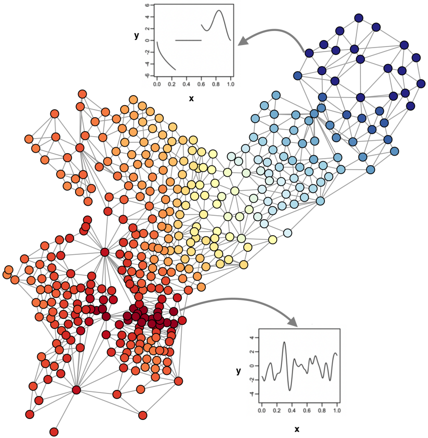


$$\mathbb{P}(\texttt{strike}) = \Phi(f(\texttt{location}, \texttt{batter}, \texttt{pitcher}, \texttt{catcher}, \texttt{umpire}))$$

where f is approximated by sum-of-trees

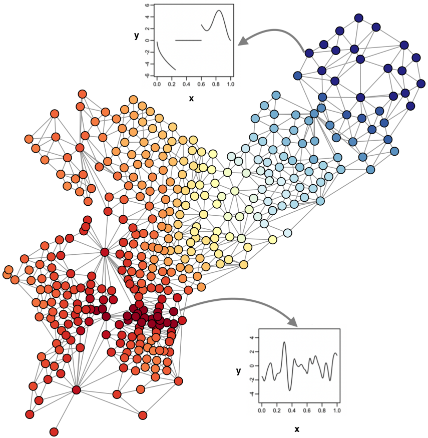
Problem 2: Network-indexed regression

- Observe t_i pairs at vertex i: $(\boldsymbol{x}_{i1}, y_{i1}), \cdots (\boldsymbol{x}_{it_i}, y_{it_i})$
- $\mathbb{E}[y|\boldsymbol{x}]$ may vary across network



Problem 2: Network-indexed regression

- Observe t_i pairs at vertex i: $(\boldsymbol{x}_{i1}, y_{i1}), \cdots (\boldsymbol{x}_{it_i}, y_{it_i})$
- $\mathbb{E}[y|\boldsymbol{x}]$ may vary across network
- Idea: vertex label as covariate
- $y_{it} \sim \mathcal{N}(f(\boldsymbol{x}_{it}, i), \sigma^2)$
- Network smoothness: for all $\boldsymbol{x}$, $f(\boldsymbol{x}, i) \approx f(\boldsymbol{x}, j)$ whenever $i \sim j$
- Can BART learn $f(\boldsymbol{x}, i)$?



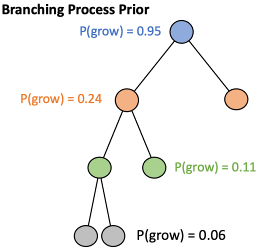
Common feature: categorical variables

- Pitch framing problem:
  - Represent batter, catcher, pitcher, and umpire as categorical predictors
  - Each year: 900+ batters, 100+ catchers, 800+ pitchers, 100+ umpires
- Network-indexed regression: vertex label as categorical predictor
- Default practice: introduce many 0/1 dummy variables
  - One dummy variable per level of each categorical predictor
  - Pitch framing: requires a massive (380000×2000) design matrix
  - Network-indexed regression: dummy variables lose adjacency structure

Common feature: categorical variables

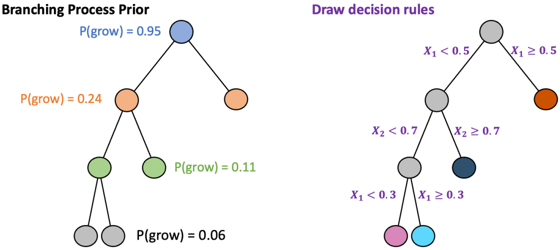
- Pitch framing problem:
  - Represent batter, catcher, pitcher, and umpire as categorical predictors
  - Each year: 900+ batters, 100+ catchers, 800+ pitchers, 100+ umpires
- Network-indexed regression: vertex label as categorical predictor
- Default practice: introduce many 0/1 dummy variables
  - One dummy variable per level of each categorical predictor
  - Pitch framing: requires a massive (380000×2000) design matrix
  - Network-indexed regression: dummy variables lose adjacency structure
- This talk: new regression tree prior that
  - Obviates the need for binary dummy variables
  - Permits splitting on network-structured categorical predictors
  - More flexibly "borrow strength" across categorical levels
  - Is available in the **flexBART** package (on GitHub)

BART's default regression tree prior



1. Generate tree topology by simulating a branching process
   - Node has 0 children (terminal) or 2 children (non-terminal)
   - $\mathbb{P}(\text{node at depth } d \text{ is non-terminal}) = 0.95(1+d)^{-2}$

BART's default regression tree prior



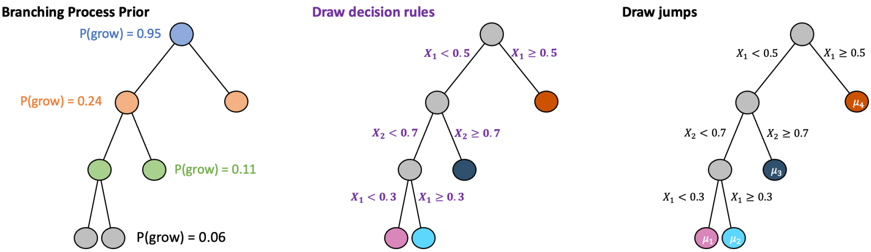
1. Generate tree topology by simulating a branching process
   - Node has 0 children (terminal) or 2 children (non-terminal)
   - $\mathbb{P}(\text{node at depth } d \text{ is non-terminal}) = 0.95(1+d)^{-2}$
2. Draw decision rule $\{X_v < c\}$ at each non-terminal node η

(2a) Pick a random variable index v

(2b) Uniformly draw $c \in \mathcal{A}$, interval of values of X_v available at η

- $\mathcal{A}$ determined by decision rules at ancestors of η in tree

BART's default regression tree prior

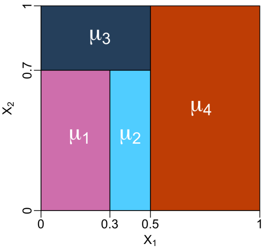


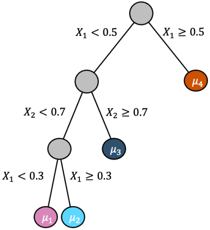
1. Generate tree topology by simulating a branching process
   - Node has 0 children (terminal) or 2 children (non-terminal)
   - $\mathbb{P}(\text{node at depth } d \text{ is non-terminal}) = 0.95(1+d)^{-2}$
2. Draw decision rule $\{X_v < c\}$ at each non-terminal node η

   (2a) Pick a random variable index v

   (2b) Uniformly draw $c \in \mathcal{A}$, interval of values of X_v available at η
   - $\mathcal{A}$ determined by decision rules at ancestors of η in tree
3. Draw jumps $\mu_\ell \sim \mathcal{N}(0, \tau^2/M)$

Regression trees & partitions of continuous space



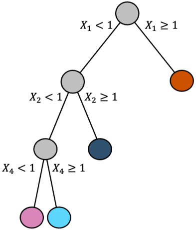


- Decision tree partitions $\mathbb{R}^p$ into axis-parallel rectangles
- Decision tree prior induces prior over these partitions

☺ **Every** such partition representable w/ a binary decision tree

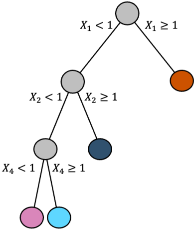
☺ Positive (if potentially tiny) prior probability on **every** such partition

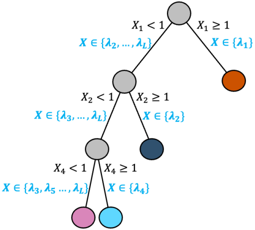
Binary dummies & partitions of categorical levels



- Consider a single categorical predictor w/ L levels: $X \in \{\lambda_1, \ldots, \lambda_L\}$
- Introduce L binary dummy variables: $X_\ell = \mathbb{1}(X = \lambda_\ell)$.

Binary dummies & partitions of categorical levels





- Consider a single categorical predictor w/ L levels: $X \in \{\lambda_1, \ldots, \lambda_L\}$
- Introduce L binary dummy variables: $X_\ell = \mathbb{1}(X = \lambda_\ell)$.
- Decision trees w/ dummies can form $2^L - L$ partitions like

$$\overbrace{\{\lambda_1\} \cup \cdots \{\lambda_k\}}^{k \text{ singletons}} \cup \overbrace{\{\lambda_{k+1}, \ldots, \lambda_L\}}^{\text{set with } L-k \text{ elements}}$$

\# partitions (Bell number): $B_L \gg (L/2)^{L/2} \gg 2^L - L$

Prior heavily restricts how trees "borrow strength" across levels

A new regression tree prior

- Use the same branching process for tree topology & prior for jumps
- New prior for decision rule $\{X_v \in \mathcal{C}\}$ at each non-terminal node η
- Important: do not convert categorical X_v's into binary dummies
  (2a) Pick a random variable index v
  (2b) Draw random subset $\mathcal{C} \subset \mathcal{A}$, set of values of X_v available at η

A new regression tree prior

- Use the same branching process for tree topology & prior for jumps
- New prior for decision rule $\{X_v \in \mathcal{C}\}$ at each non-terminal node η
- Important: do not convert categorical X_v's into binary dummies

(2a) Pick a random variable index v

(2b) Draw random subset $\mathcal{C} \subset \mathcal{A}$, set of values of X_v available at η

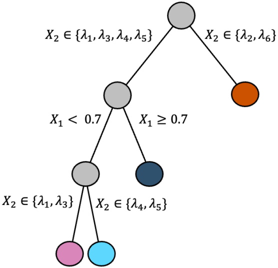
- If X_v continuous: set of available values $\mathcal{A}$ is an interval
  - Draw $c \in \mathcal{A}$ & set $\mathcal{C} = (-\infty, c) \cap \mathcal{A}$
  - I.e. continuous decision rules drawn exactly as before

A new regression tree prior

- Use the same branching process for tree topology & prior for jumps
- New prior for decision rule $\{X_v \in \mathcal{C}\}$ at each non-terminal node η
- Important: do not convert categorical X_v's into binary dummies
  (2a) Pick a random variable index v
  (2b) Draw random subset $\mathcal{C} \subset \mathcal{A}$, set of values of X_v available at η
- If X_v continuous: set of available values $\mathcal{A}$ is an interval
  - Draw $c \in \mathcal{A}$ & set $\mathcal{C} = (-\infty, c) \cap \mathcal{A}$
  - I.e. continuous decision rules drawn exactly as before
- If X_v categorical: set of available values $\mathcal{A}$ is a discrete set
  - If X_v unstructured: $\mathbb{P}(a \in \mathcal{C}) = 1/2$ independently for each $a \in \mathcal{A}$
  - If X_v network structured: randomly partition $\mathcal{A}$ but respect adjacency

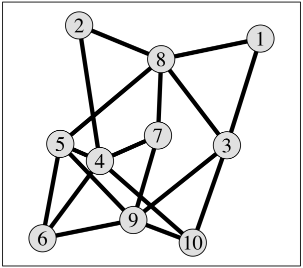
Example

- Continuous $X_1 \in [0, 1]$
- Categorical $X_2 \in \{\lambda_1, \ldots, \lambda_6\}$
- To draw decision rule $\{X_v \in \mathcal{C}\}$
  (i) Draw splitting variable index v
  (ii) Compute $\mathcal{A}$, set of available values of X_v
  (iii) Divide $\mathcal{A}$ into two parts



Splitting on a network-structured variable

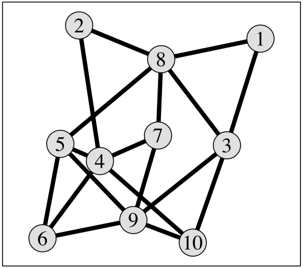
Graph of available vertices



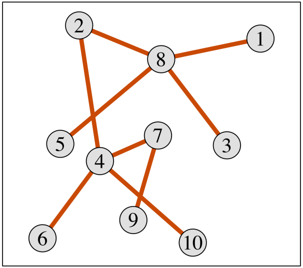
- Look at the *subgraph* induced by elements of available levels $\mathcal{A}$
- Idea: partition graph into two connected subsets

Splitting on a network-structured variable

Graph of available vertices

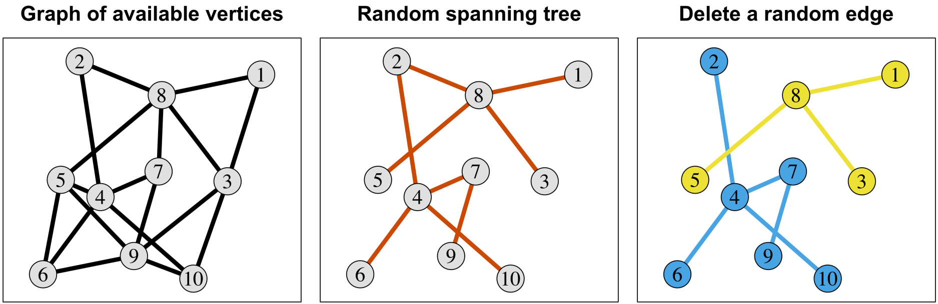


Random spanning tree



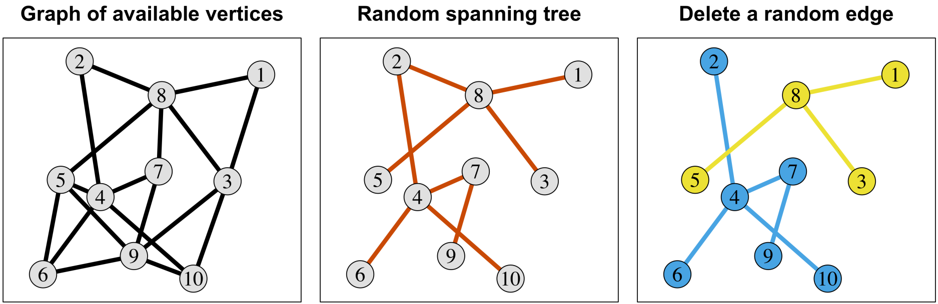
- Look at the *subgraph* induced by elements of available levels $\mathcal{A}$
- Idea: partition graph into two connected subsets
  (1) Uniformly draw a random spanning tree w/ Wilson's algorithm

Splitting on a network-structured variable



- Look at the *subgraph* induced by elements of available levels $\mathcal{A}$
- Idea: partition graph into two connected subsets
  (1) Uniformly draw a random spanning tree w/ Wilson's algorithm
  (2) Delete a uniform edge from spanning tree (e.g. (2,8))
  (3) Set $\mathcal{C}$ to be set of vertices in one component of cut graph

Splitting on a network-structured variable



- Look at the *subgraph* induced by elements of available levels $\mathcal{A}$
- Idea: partition graph into two connected subsets
  (1) Uniformly draw a random spanning tree w/ Wilson's algorithm
  (2) Delete a uniform edge from spanning tree (e.g. (2,8))
  (3) Set $\mathcal{C}$ to be set of vertices in one component of cut graph

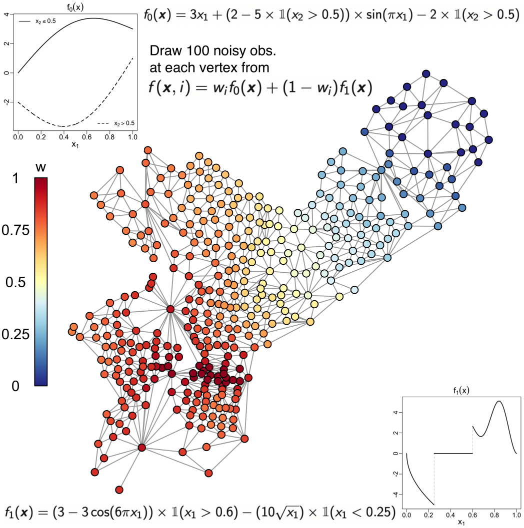
☺ Recursively applying process can produce **all** network partitions into connected components

Results: pitch framing

- 10-fold cross-validation w/ all data from 2019 ($n \approx 380,000$)
- Compared probit implementations in **flexBART** and **BART**
- Compared to **BART**, **flexBART**
  - Produced more accurate out-of-sample predictions
  - Was considerably faster (1.5 vs 18 hours for 2000 iterations)
  - Had a smaller memory footprint (3 vs 17 GB)
- **BART** repeatedly evaluates regression trees at every observed input
  - Results in several redundant computations per iteration
  - **flexBART** saves & updates map of observation to tree leaves
- **BART** introduced ~ 2000 dummy variables
  - 5 GB to store the $n \times 2000$ matrix
  - 2.5 GB to save matrix of posterior samples

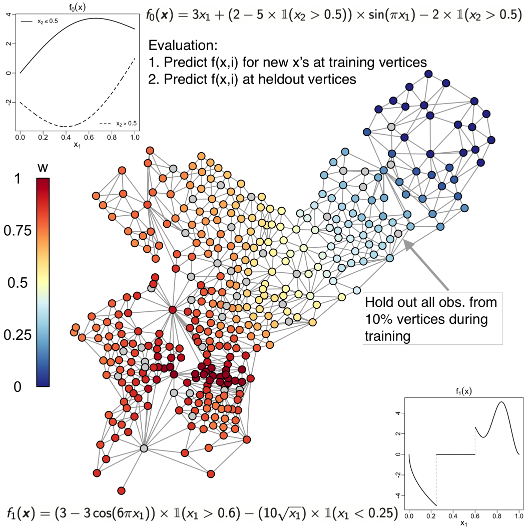
Semi-synthetic network-index regression

- For $i = 1, \ldots, n$ and $t = 1, \ldots 100$
  - $\boldsymbol{x}_{it} \sim \mathcal{U}([0,1]^{10})$
  - $y_{it} \sim \mathcal{N}(f(\boldsymbol{x}_{it}, i), 1)$
- Hold out $(\boldsymbol{x}_{it}, y_{it})$'s from 10% of vertices during training



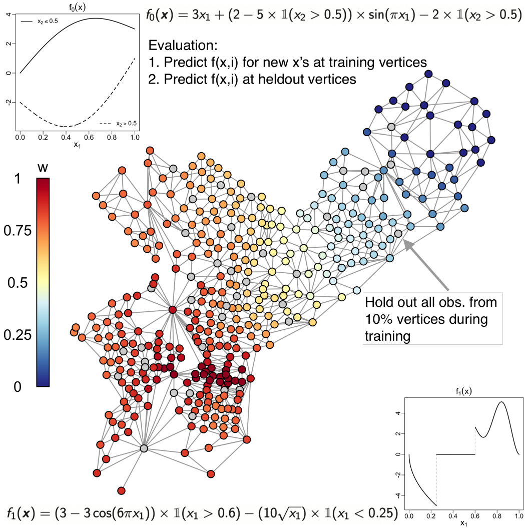
Semi-synthetic network-index regression

- For $i = 1, \ldots, n$ and $t = 1, \ldots 100$
  - $\boldsymbol{x}_{it} \sim \mathcal{U}([0,1]^{10})$
  - $y_{it} \sim \mathcal{N}(f(\boldsymbol{x}_{it}, i), 1)$
- Hold out $(\boldsymbol{x}_{it}, y_{it})$'s from 10% of vertices during training



Semi-synthetic network-index regression

- For $i = 1, \ldots, n$ and $t = 1, \ldots 100$
  - $\boldsymbol{x}_{it} \sim \mathcal{U}([0,1]^{10})$
  - $y_{it} \sim \mathcal{N}(f(\boldsymbol{x}_{it}, i), 1)$
- Hold out $(\boldsymbol{x}_{it}, y_{it})$'s from 10% of vertices during training
- ☺ Out-of-sample RMSE ratio (**BART**/**flexBART**)
  - Training vertices: 2.9
  - Heldout vertices: 3.5
- ☺ **flexBART** still faster! 38 vs 93 min.



$f_0(\boldsymbol{x}) = 3x_1 + (2 - 5 \times \mathbb{1}(x_2 > 0.5)) \times \sin(\pi x_1) - 2 \times \mathbb{1}(x_2 > 0.5)$

Evaluation:
1. Predict f(x,i) for new x's at training vertices
2. Predict f(x,i) at heldout vertices

$f_1(\boldsymbol{x}) = (3 - 3\cos(6\pi x_1)) \times \mathbb{1}(x_1 > 0.6) - (10\sqrt{x_1}) \times \mathbb{1}(x_1 < 0.25)$

Takeaways & next steps

- New decision rule prior obviates the need to use 0/1 dummy variable
- Network splitting prior induces a new kernel over graphs
- Experimenting w/ alternative network splitting procedures
- Other BART extensions can use the new decision rule prior

Thanks, y'all!

Email: sameer.deshpande@wisc.edu
Package: https://github.com/skdeshpande91/flexBART
Website: https://skdeshpande91.github.io