FIRE: An Optimization Approach for Fast Interpretable Rule Extraction

Alessandro Morosini & Siyu Zhang

March 8, 2025

Outline

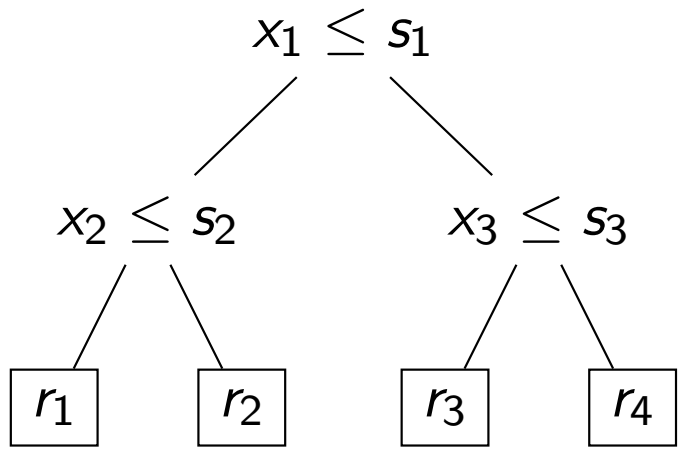
Decision Trees

Decision Trees:

- Hierarchical models that partition data based on features
- Each internal node represents a split on a feature: $(x_i \leq s_i)$
- Each leaf node represents a prediction
- Decision rule: a path from root to leaf

Training:

- Find splits that minimize impurity/error
- Common algorithms: CART, OCT..



Decision tree with 4 rules

Decision Trees: Mathematical Representation

- Consider a decision tree Γ fit on data matrix $X \in \mathbb{R}^{N \times P}$
- The tree has R leaf nodes, each with prediction value v_j for $j \in [R]$
- We define a mapping matrix $M \in \mathbb{R}^{N \times R}$ whose (i, j)-th entry is:

$$M_{ij} = \begin{cases} v_j & \text{if data point } x_i \text{ reaches leaf node } r_j \\ 0 & \text{otherwise} \end{cases}$$

- For each leaf node j, we assign a weight w_j
- Let weight vector $w \in \mathbb{R}^R$ represent these weights
- The prediction of this tree is given by Mw
- For standard decision trees, $w = \mathbf{1}$ (all weights are 1)
- When we extract rules, we optimize w to select a sparse subset of rules

Decision Rules: Example

- Recall decision tree from previous slide: 4 leaves
- Let the values of leaves be $v = [0.2, 0.5, 0.8, 1.0]$
- Assume $n = 5$ samples
- x_1 ends in leaf 1, x_2 in leaf 2, x_3 in leaf 3, x_4 in leaf 4, and x_5 in leaf 3

$$M = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0.8 & 0 \end{bmatrix}$$

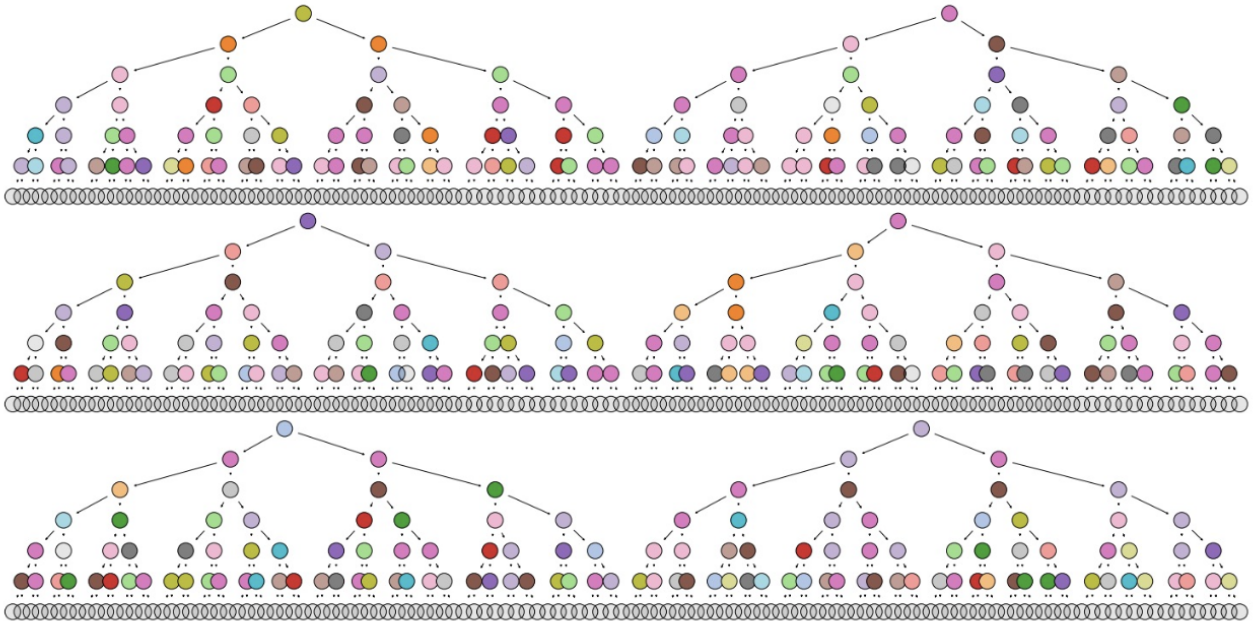
- The predictions will be Mw where $w = [w_1, w_2, w_3, w_4]^T$

$$Mw = \begin{bmatrix} 0.2 \cdot w_1 \\ 0.5 \cdot w_2 \\ 0.8 \cdot w_3 \\ 1.0 \cdot w_4 \\ 0.8 \cdot w_3 \end{bmatrix}$$

Tree Ensembles

- Simple trees are interpretable but limited in accuracy
- **Tree Ensembles**: Collections of decision trees whose predictions are combined
  - **Random Forest**: Trees trained on bootstrapped samples (bagging)
  - **XGBoost**: Sequential addition of trees (boosting)
- Powerful and versatile but become massive in size
- As ensemble grows, interpretability decreases dramatically

The Interpretability Problem



Visualization of a tree ensemble

- Ensembles may contain hundreds or thousands of trees
- Each tree adds dozens or hundreds of decision paths
- Impossible for humans to comprehend the combined model

Notation and Problem Setup

- Tree ensemble with T trees: $\{\Gamma_t(X) : t \in [T]\}$
- Each tree Γ_t has R_t leaf nodes (rules)
- Mapping matrix $M_t \in \mathbb{R}^{N \times R_t}$:

$$(M_t)_{ij} = \begin{cases} v_j & \text{if data point } x_i \text{ reaches leaf node } r_j \\ 0 & \text{otherwise} \end{cases}$$

- Total mapping matrix: $M = [M_1, M_2, \ldots, M_T] \in \mathbb{R}^{N \times R}$
- Weight vector: $w \in \mathbb{R}^R$
- Prediction: Mw

Motivation for FIRE

- Tree ensembles result in large collections of rules
- Goal: Extract a sparse but representative subset

Idea

- Enforce sparsity on w
- Enforce sharing of parent nodes

Challenges

- The number of rules scales exponentially with tree depth
- Rules from shallow ensembles are highly correlated

The FIRE Optimization Framework

$$\min_{w} f(w) + h(w, \lambda_s) + g(w, \lambda_f)$$

where:

- $f(w) = \frac{1}{2}\|y - Mw\|_2^2$ (quadratic loss)
- $h(w, \lambda_s)$ = sparsity penalty with parameter λ_s
- $g(w, \lambda_f)$ = fusion penalty with parameter λ_f

Novel Aspects

- Non-convex MCP penalty for aggressive rule selection
- Fused LASSO penalty to encourage rule fusion

Sparsity with Minimax Concave Penalty (MCP)

$$h(w, \lambda_s) = \sum_{i=1}^{R} P_\gamma(w_j, \lambda_s)$$

where $P_\gamma(w_j, \lambda_s)$ is the MCP penalty function:

$$P_\gamma(w_j, \lambda_s) = \begin{cases} \lambda_s |w_j| - \frac{w_j^2}{2\gamma} & \text{if } |w_j| \leq \lambda_s \gamma \\ \frac{1}{2}\gamma\lambda_s^2 & \text{if } |w_j| > \lambda_s \gamma \end{cases}$$

- $\gamma > 1$ controls concavity:
- Compared to LASSO (ℓ_1-penalty):
  - Behaves like ℓ_0-penalty when $\gamma \to 1^+$
  - Behaves like ℓ_1-penalty (LASSO) when $\gamma \to \infty$
  - Less bias for large coefficients
  - Better at sparse selection with correlated features

Rule Fusion with Fused LASSO

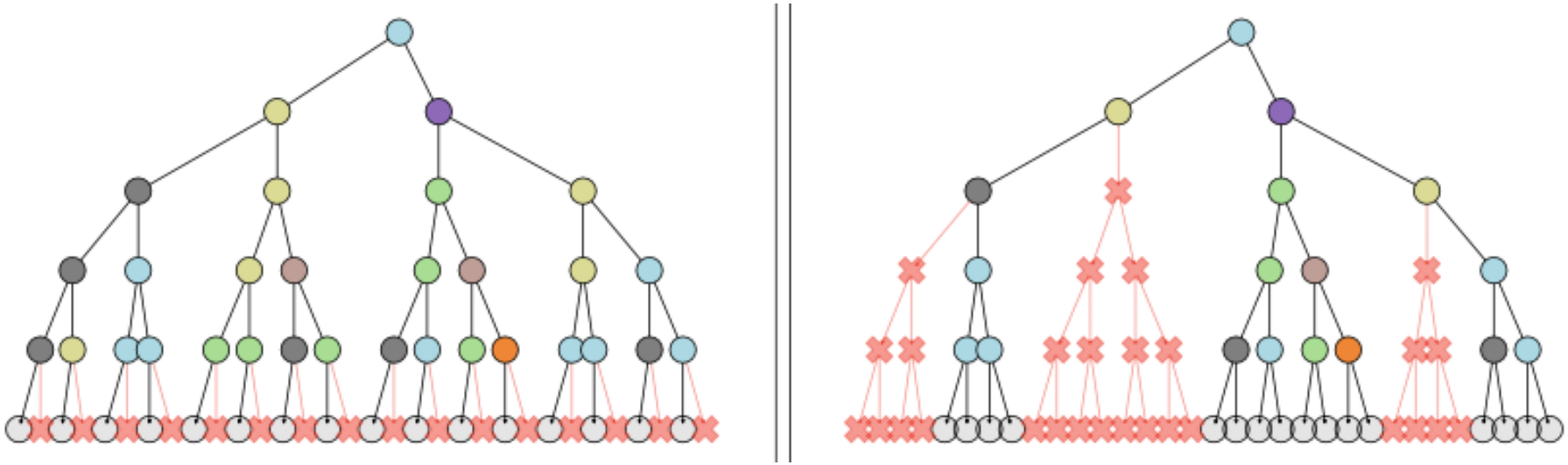
$$g(w, \lambda_f) = \lambda_f \sum_{t=1}^{T} \|D_t w_t\|_1$$

where $D_t \in \{-1, 0, 1\}^{(R_t - 1) \times R}$ is the fusion matrix such that

$$\|D_t w_t\|_1 = \sum_{j=2}^{R_t} |(w_t)_j - (w_t)_{j-1}|$$

- Penalizes differences between adjacent weights in each tree
- Encourages contiguous sections of selected/pruned rules

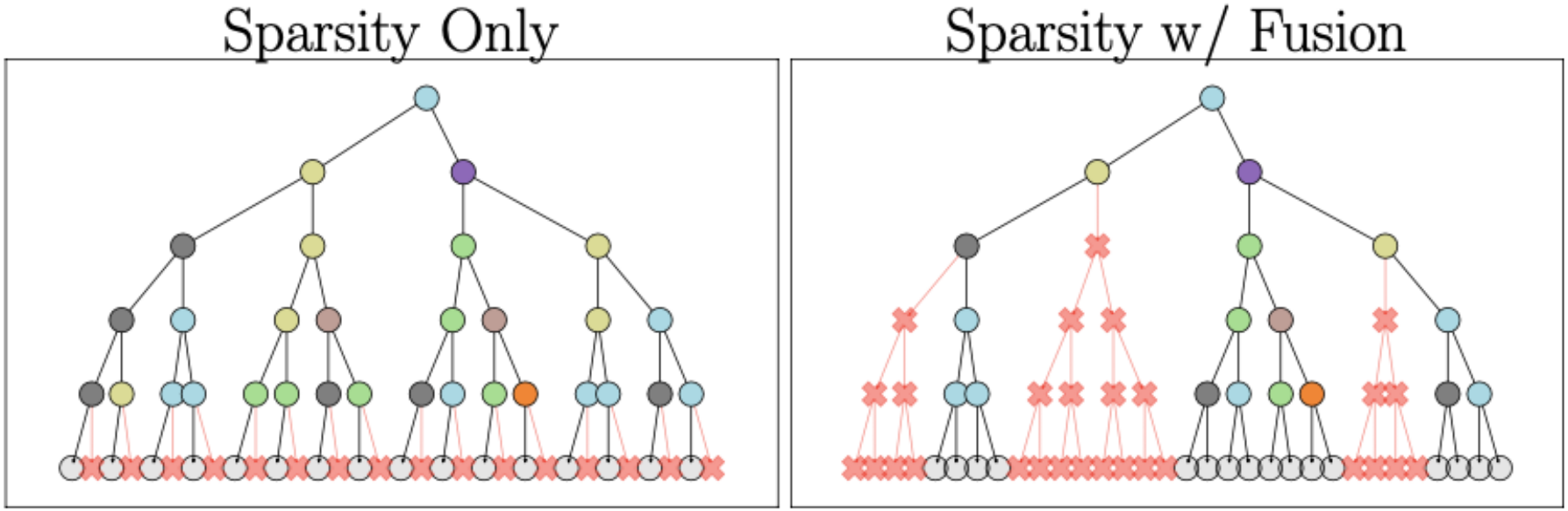
Question



Tree A vs. Tree B

- Which tree is easier to interpret?
- Which one is using fusion?

Answer



Comparison: Sparsity Only vs. Sparsity with Fusion

- Both trees have 16 leaf nodes
- But sparsity with fusion leads to 44% fewer internal nodes
- Rules share common antecedents, making them easier to interpret

Optimization Challenges

- Problem scale: Hundreds of thousands of variables
- Non-convexity: MCP penalty is non-convex
- High correlation among rules

Traditional Solvers Are Inadequate

- Standard approaches too slow for large ensembles
- Coordinate descent methods inefficient with many variables
- Need specialized solver that exploits problem structure

Greedy Block Coordinate Descent (GBCD)

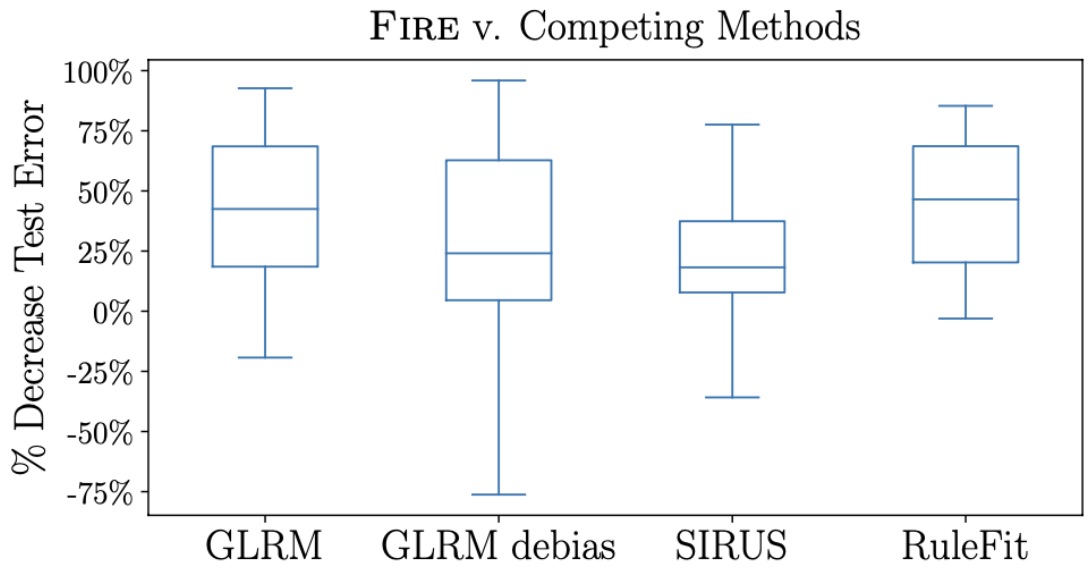
- Key insight: Exploit natural blocking structure of the problem
- Each tree forms a natural "block" of variables
- Novel algorithm components:
  - Block proximal updates with efficient solvers
  - Greedy block selection based on steepest direction
  - Closed-form solutions for block updates

Algorithm 1 GBCD Solver

1: Initialize $w = 0$
2: **repeat**
3: Find steepest direction vector d
4: Select block t with largest $\|d_t\|$
5: Update block w_t using proximal gradient
6: **until** converged

Performance Summary

- Compared against state-of-the-art methods:
  - RuleFit (LASSO-based rule extraction)
  - SIRUS (stabilized tree ensemble)
  - GLRM (column generation for rules)
- Restricted to sparse, interpretable models (< 15 rules)



Percent Decrease in Test Error compared to other methods

Using FIRE in Practice

- GitHub repository:
  `https://github.com/brianliu12437/FIREKDD2023`
- Key hyperparameters:
  - λ_s: Controls sparsity (number of rules)
  - γ: Controls concavity of MCP penalty
  - λ_f: Controls fusion strength
- Example usage:
  `https://github.com/AlessandroMorosini/fire-presentation`

Open Questions and Future Work

- Extending FIRE to other ensemble types (e.g., boosting)
- Incorporating domain knowledge into rule selection
- Developing interactive tools for rule exploration
- Theoretical guarantees on rule selection quality

Summary

- FIRE is a novel optimization framework for rule extraction
- Key innovations:
  - MCP penalty for aggressive rule selection
  - Fusion penalty for improved interpretability
  - Specialized GBCD solver for efficiency
- Results:
  - Better performance than state-of-the-art methods
  - More interpretable rule ensembles
  - Computationally efficient, scaling to large problems

References

Liu, B., & Mazumder, R. (2023).
FIRE: An Optimization Approach for Fast Interpretable Rule Extraction.
Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23).