PaloBoost

An Overfitting-robust TreeBoost with Out-of-Bag Sample Regularization Technique

https://arxiv.org/abs/1807.08383

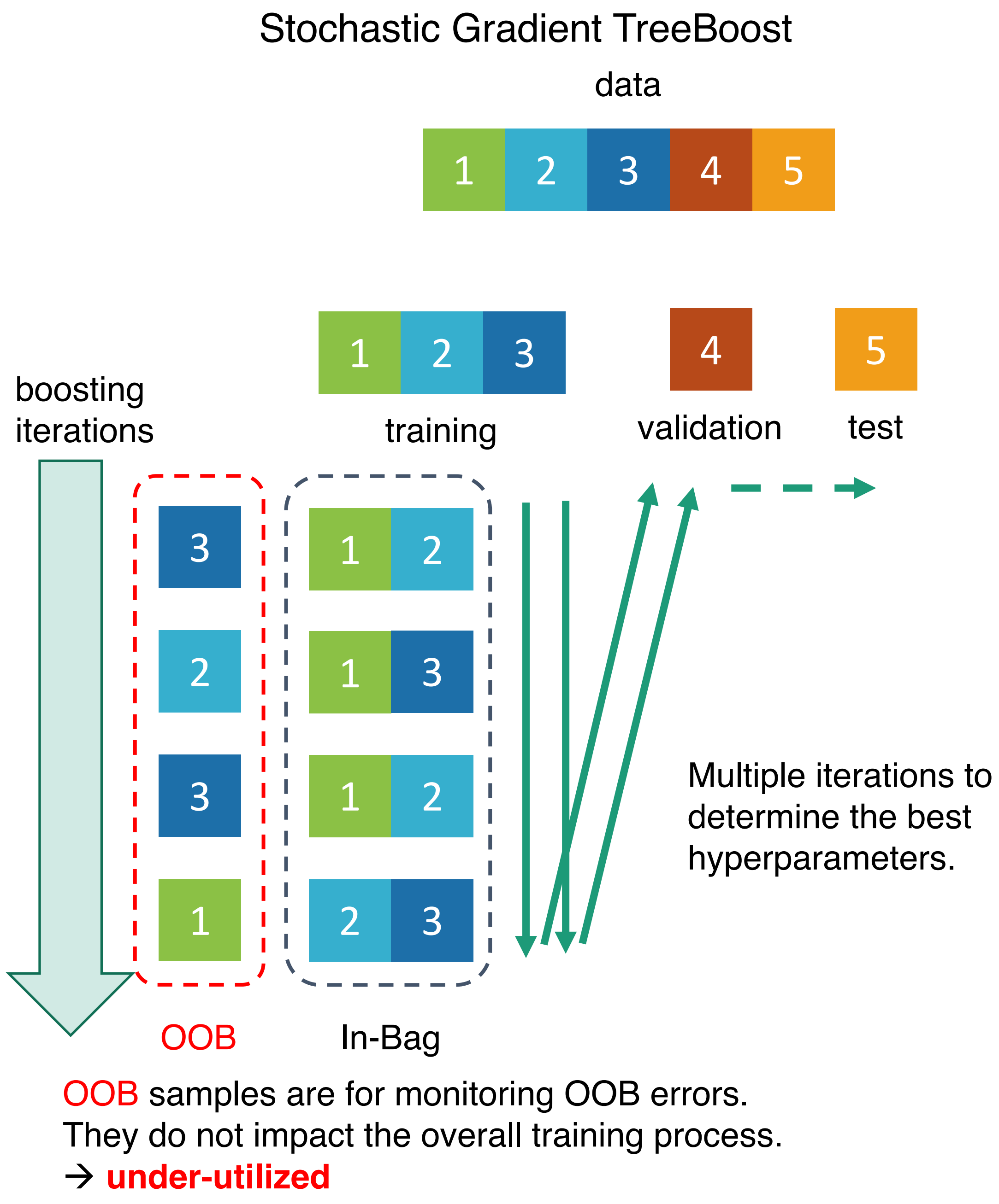
Yubin Park and Joyce Ho

Motivation

- Stochastic Graident TreeBoost is a powerful algorithm
  - Many winning solutions in various data science challenges
  - Fast training speed compared other algorithms
  - Can handle various data types including missing values
- BUT, the algorithm can be even ***more powerful*** IF…
  - It is less sensitive to hyperparameters, such as learning rate, tree depth, etc.
  - And at the same time, it provides robust performance without overfitting
- The question is: Can we build such an algorithm?

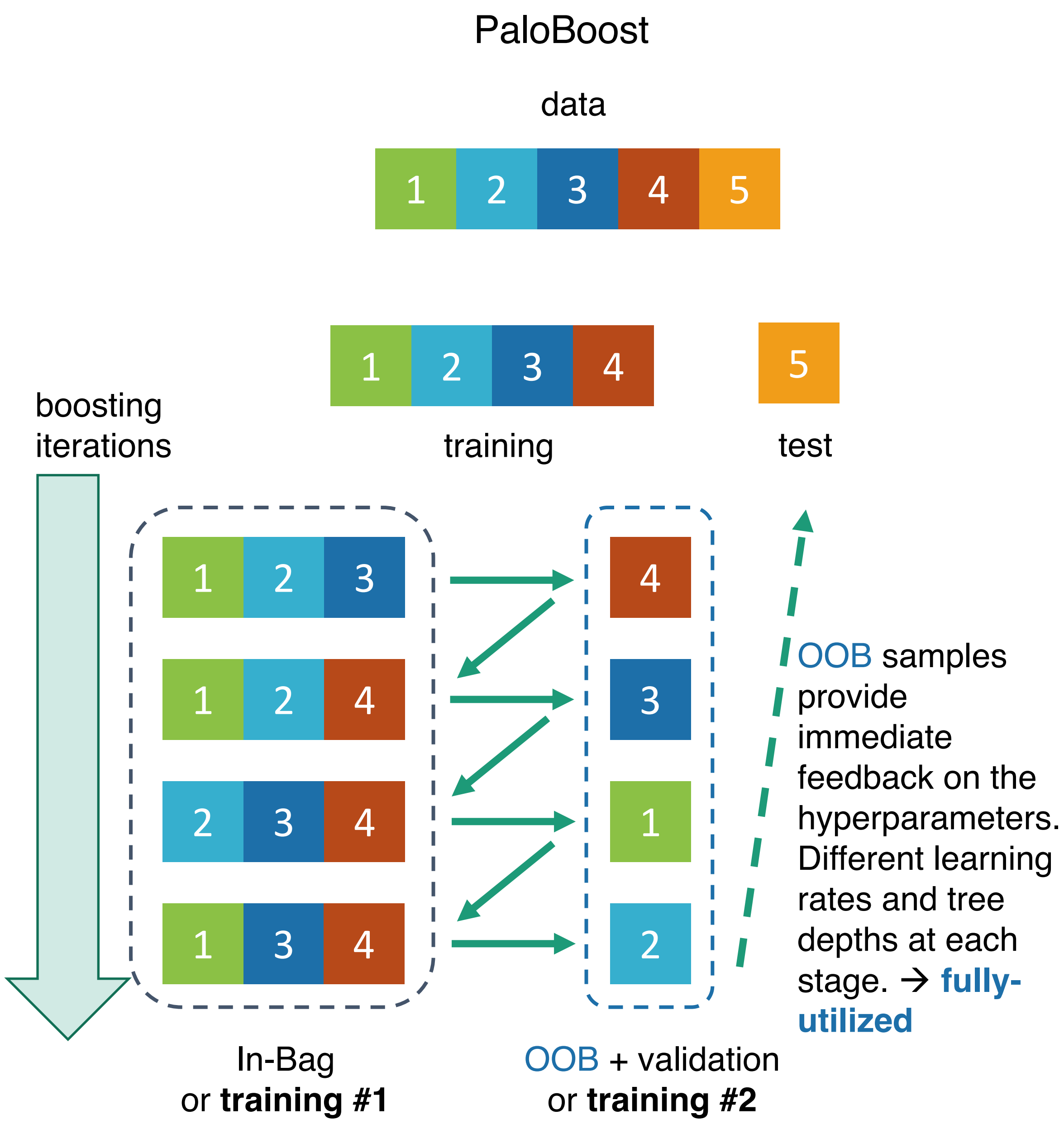
Out-of-Bag Samples? (1)

- Out-of-Bag (OOB) samples
  - The samples that are not included in the subsampling process
  - Available in many subsampling-based algorithms such as Random Forests
- In Stochastic Gradient TreeBoost,
  - OOB samples are often used to estimate cross-validation errors
  - They can be used for early-stopping
  - However, the OOB errors do not affect the actual training process much
- We think OOB samples can provide more values than this



Out-of-Bag Samples? (2)

- PaloBoost utilizes the under-utilized out-of-bag samples
- Gradient-Aware Pruning
  - Prune a decision tree if its gradient steps do not decrease OOB errors
- Adaptive Learning Rates
  - Find the optimal learning rates for the gradient steps w.r.t. OOB errors



PaloBoost: Adaptive Learning Rate

- PaloBoost sets different learning rates for each tree leaf
- This allows closed form solutions for the optimal learning rates
  - Bernoulli distribution:

$$\nu_j^* = \text{argmin}_\nu \sum_i^{L_j} \log(1 + \exp(\hat{y}_i + \nu\gamma_j)) - y_i(\hat{y}_i + \nu\gamma_j)$$

$$= \log\left(\frac{\sum_i^{L_j} y_i}{\sum_i^{L_j}(1 - y_i)\exp(\hat{y}_i)}\right)/\gamma_j$$

  - Gaussian distribution:

$$\nu_j^* = \text{argmin}_\nu \sum_i^{L_j}(y_i - (\hat{y}_i + \nu\gamma_j))^2$$

$$= \frac{\sum_{i=1}^{L_j}(y_i - \hat{y}_i)}{\gamma_j L_j}$$

ALGORITHM 5: PaloBoost

$F_0 = \arg\min_\beta \sum_{i=1}^{N} L(y_i, \beta)$

for $m \leftarrow 1$ ***to*** M **do**

$\{y_i, \mathbf{x}_i\}_1^{N'} = \text{Subsample}(\{y_i, \mathbf{x}_i\}_1^N, \text{rate} = q)$

$z_i = -\frac{\partial}{\partial F(\mathbf{x}_i)} L(y_i, F(\mathbf{x}_i))\Big|_{F=F_{m-1}}$, for $i = 1, \cdots, N'$

$\{R_{jm}\}_1^J = \text{RegressionTree}(\{z_i, \mathbf{x}_i\}_i^{N'})$

$\gamma_{jm} = \arg\min_\gamma \sum_{\mathbf{x}_i \in R_{jm}} L(y_i, F_{m-1}(\mathbf{x}_i) + \gamma)$, for $j = 1, \cdots, J$

$\{R_{jm}, \gamma_{jm}\}_1^{J'} = \text{Gradient-Aware-Pruning}(\{R_{jm}, \gamma_{jm}\}_1^J, \nu_{max})$

$\nu_{jm} = \text{Learning-Rate-Adjustment}(\gamma_{jm}, \nu_{max})$, for $j = 1, \cdots, J'$

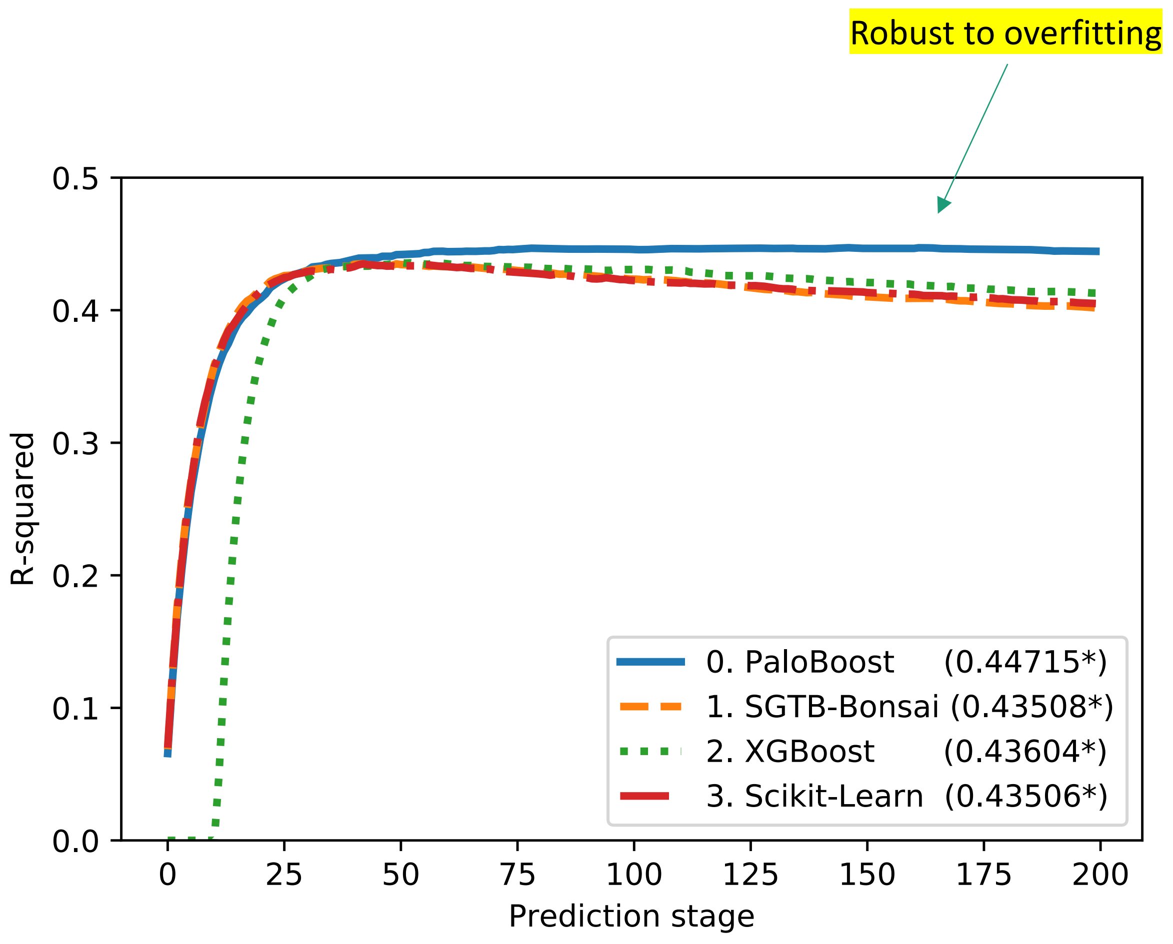
$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \nu_{jm}\gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm})$

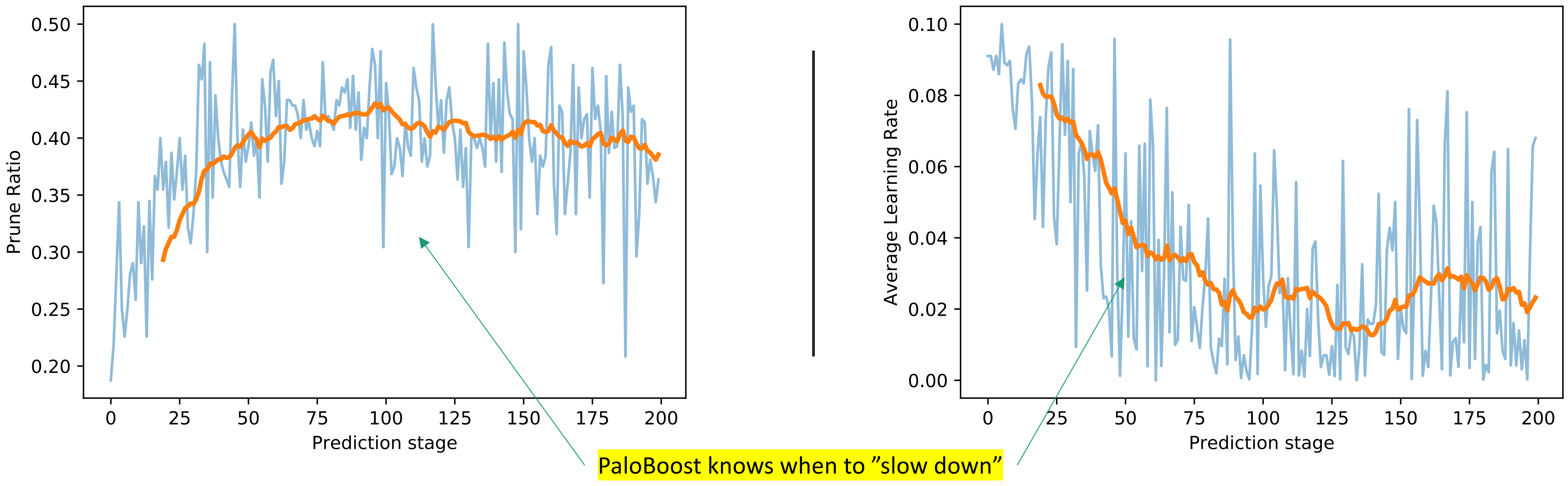
PaloBoost: Algorithm

For more information, please see:

Experimental Results: Predictive Performance

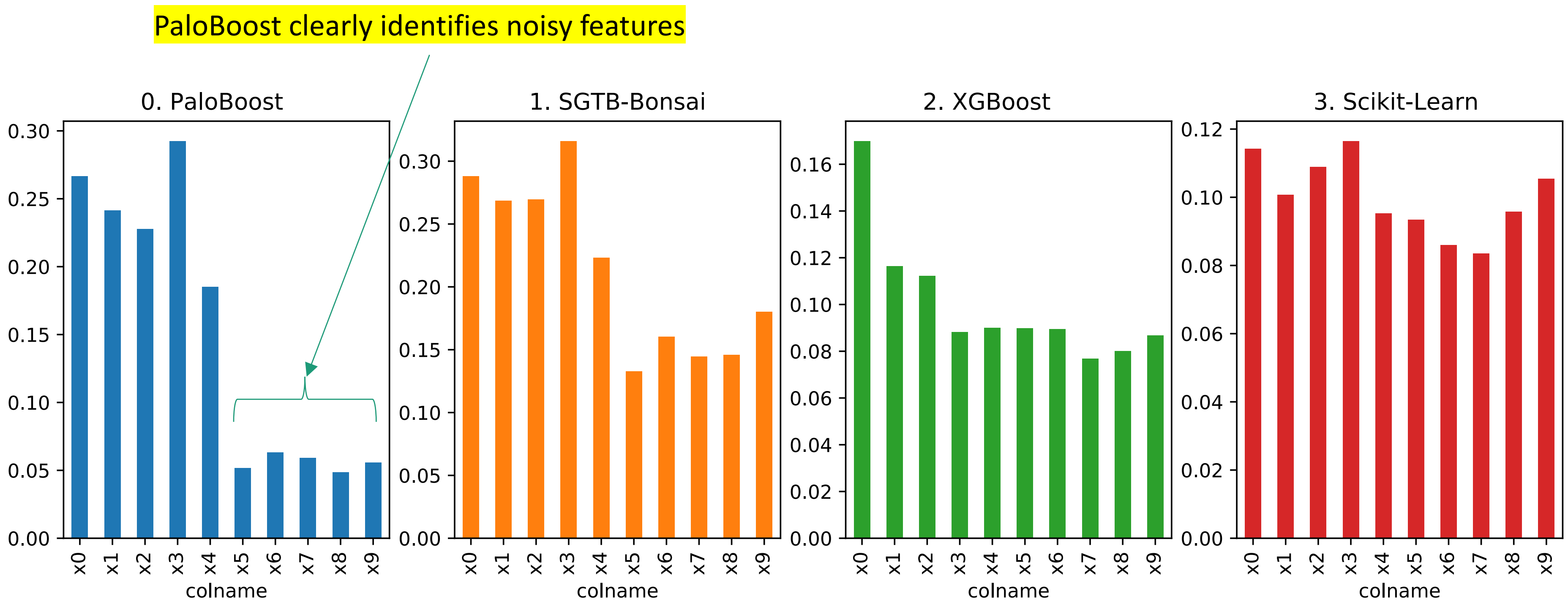
Friedman's Simulated Data in [“Multivariate adaptive regression splines”]





Experimental Results: OOB Regularizations

Gradient-Aware Pruning (left) and Adaptive Learning Rate (right)



Experimental Results: Feature Importances

x5-x9 are noisy features. In theory, they should have zero importance.

Discussions

- PaloBoost automatically adapts to mitigate overfitting by knowing when it needs to "slow down"
- Moreover, PaloBoost's importance estimates can accurately capture the true importances
- Future improvements:
  - Removing low learning rate trees?
  - Combining with learning rate schedule approaches?
  - Reducing the impact of "max" learning rate and tree depth?

Software

- Available in the Bonsai-DT framework
  - Project Website: https://yubin-park.github.io/bonsai-dt/
  - PaloBoost: https://github.com/yubin-park/bonsai-dt/blob/master/bonsai/ensemble/paloboost.py
  - Script for the Experimental Results: https://github.com/yubin-park/bonsai-dt/blob/master/research/paloboost.ipynb
  - Paper link: https://arxiv.org/abs/1807.08383