Tree-Based Recursive Partitioning for Finding the Best Treatment

Eli Ben-Michael
Department of Statistics, UC Berkeley

Motivation

When designing an experiment we must choose treatments to assign to subjects. We do not know *a priori* the effect of any treatment and in many settings the treatments lie in a continuous, multidimensional space. Randomized experimentation is the standard for evaluating the effect of any treatment; however, due to budget, computational, or time constraints, we often want to find the best treatment with as few experiments as possible. **Given a budget constraint, how do we decide which treatments to give and what experiments to run?**
We consider this problem as an instance of a constrained optimization problem where:

- The objective is expensive to evaluate and possibly non-convex
- We are restricted to noisy zeroth-order information
- We have a limited budget for function queries

This problem is similar to hyper-parameter optimization in machine learning, for which [1] suggests **Bayesian Optimization**: modeling the objective as a Gaussian Process and optimizing a cheaper surrogate function.

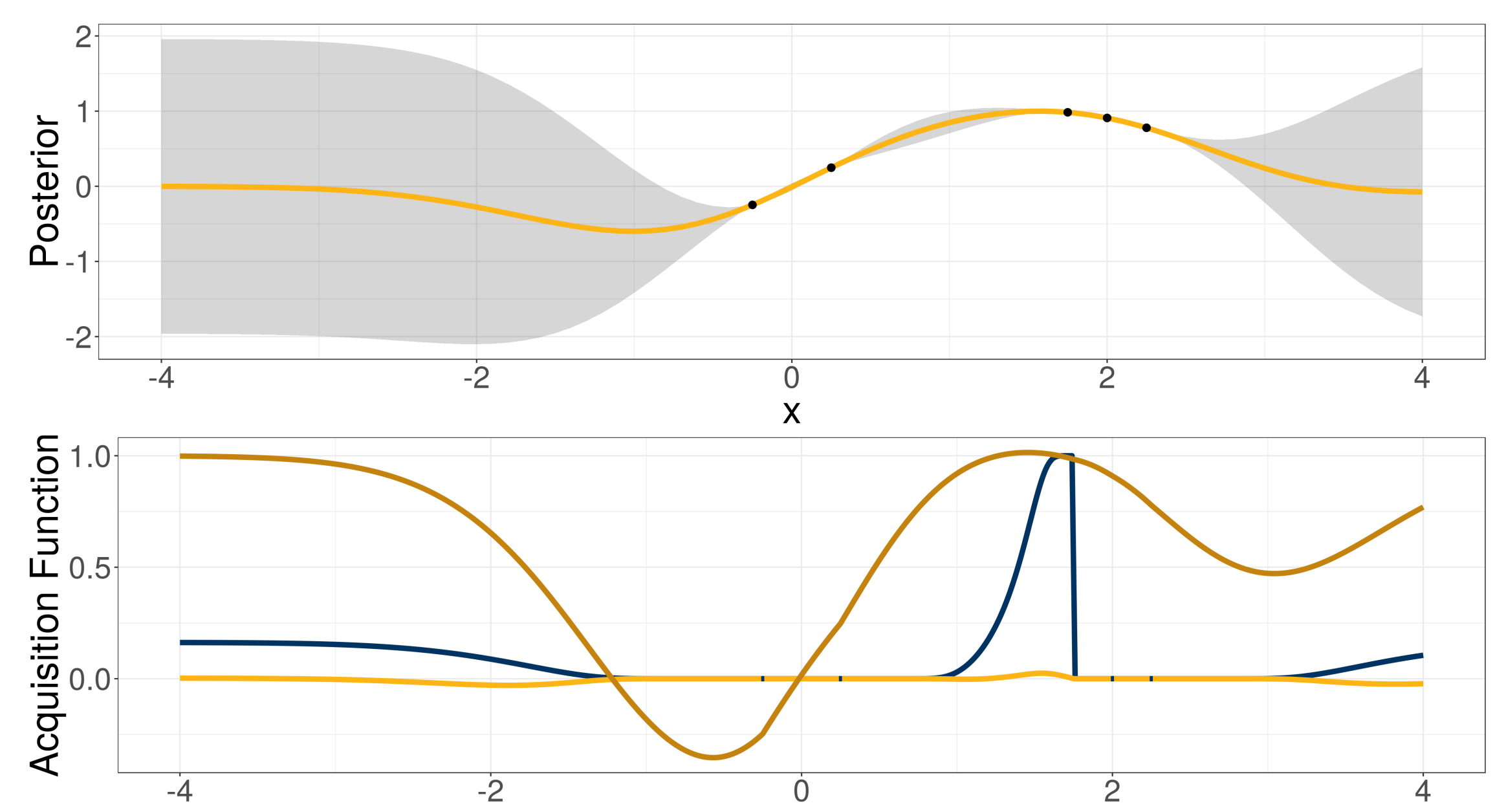


Figure: GP posterior along with probability of improvement (blue), expected improvement (gold), and upper confidence bound (brown)

[2] proposes **Hyperband**, a bandit-based approach to random search which builds off of **Sequential Halving** from [3, 4]. We fuse modeling and bandit-based ideas, and propose two algorithms which take advantage of the structure of random experiments and recursively partition the search space, producing finer partitions near possibly optimal points.

Two Tree-Based Partition Algorithms

Algorithm 1 SequentialTree

```
1: procedure SequentialTree(Box constrained space $\mathcal{X} \subset \mathbb{R}^d$, budget $T$, $\eta$, $m$)
2:     Assign $\mathcal{P} = \{\mathcal{X}\}$ and n_nodes $= m$
3:     for $r = 1 \dots \lceil \log_\eta(m) \rceil$ do
4:         Assign the number of pulls per element of $\mathcal{P}$, $n_r = \left\lfloor \frac{T}{|\mathcal{P}| \lceil \log_\eta(m) \rceil} \right\rfloor$
5:         Assign $\mathcal{A}_r = \emptyset$ and $\hat{B}_r = \emptyset$
6:         for $p \in \mathcal{P}$ do
7:             Remove $p$ from $\mathcal{P}$
8:             Sample $\{x_{p1}, \dots, x_{pn_r}\}$ independently and uniformly over $p$
9:             Let $\{y_{p1}, \dots, y_{pn_r}\}$ be the result of querying the function at these points
10:            Train a decision tree with n_nodes nodes, $t_p = \text{DecisionTree}(y_p \sim X_p)$
11:            Add the n_nodes elements of the partition of $p$ defined by $t_p$ to $\mathcal{A}_r$
12:            Add the prediction from $t_p$ for each element to $\hat{B}_r$.
13:        if $|\mathcal{P}| > \eta^2$ then
14:            Set $\mathcal{P}$ to be the elements of $\mathcal{A}_r$ with predictions in the top $\frac{|\mathcal{P}|}{\eta^2}$ of $\hat{B}_r$
15:        else
16:            Let $\hat{p}$ be the element of $\mathcal{P}$ with the best prediction
17:            return MidPoint($\hat{p}$)
18:        n_nodes $= \eta$
```

Algorithm 2 PartitionTree

```
1: procedure PartitionTree(Box constrained space $\mathcal{X} \subset \mathbb{R}^d$, budget $T$, $\eta$, $R$, $k$)
2:     Assign $\mathcal{P} = \{\mathcal{X}\}$, and $m = \eta^2$
3:     for $r = 1, \dots, R$ do
4:         The number of pulls per element of $\mathcal{P}$ $n_r = \left\lfloor \frac{T}{|\mathcal{P}| R} \right\rfloor$
5:         Assign $\mathcal{A}_r = \emptyset$ and $\hat{B}_r = \emptyset$
6:         for $p \in \mathcal{P}$ do
7:             Remove $p$ from $\mathcal{P}$
8:             Sample $\{x_{p1}, \dots, x_{pn_r}\}$ independently and uniformly over $p$
9:             Let $\{y_{p1}, \dots, y_{pn_r}\}$ be the result of querying the function at these points
10:            Train a decision tree with $m$ nodes, $t_p = \text{DecisionTree}(y_p \sim X_p)$
11:            Train a random forest with $k$ trees and $m$ nodes per tree
12:            $rf_p = \text{RandomForest}(y_p \sim X_p)$
13:            Add the $m$ elements of the partition of $p$ defined by $t_p$ to $\mathcal{A}_r$
14:            Add the prediction from $rf_p$ for the mid point of each element to $\hat{B}_r$.
15:        Set $\mathcal{P}$ to be the elements of $\mathcal{A}_r$ with predictions in the top $\frac{|\mathcal{P}|}{\eta}$ of $\hat{B}_r$
16:        Let $\hat{p}$ be the element of $\mathcal{P}$ with the best prediction
17:        $m = \eta$
18:    return MidPoint($\hat{p}$)
```

Empirical Evaluation

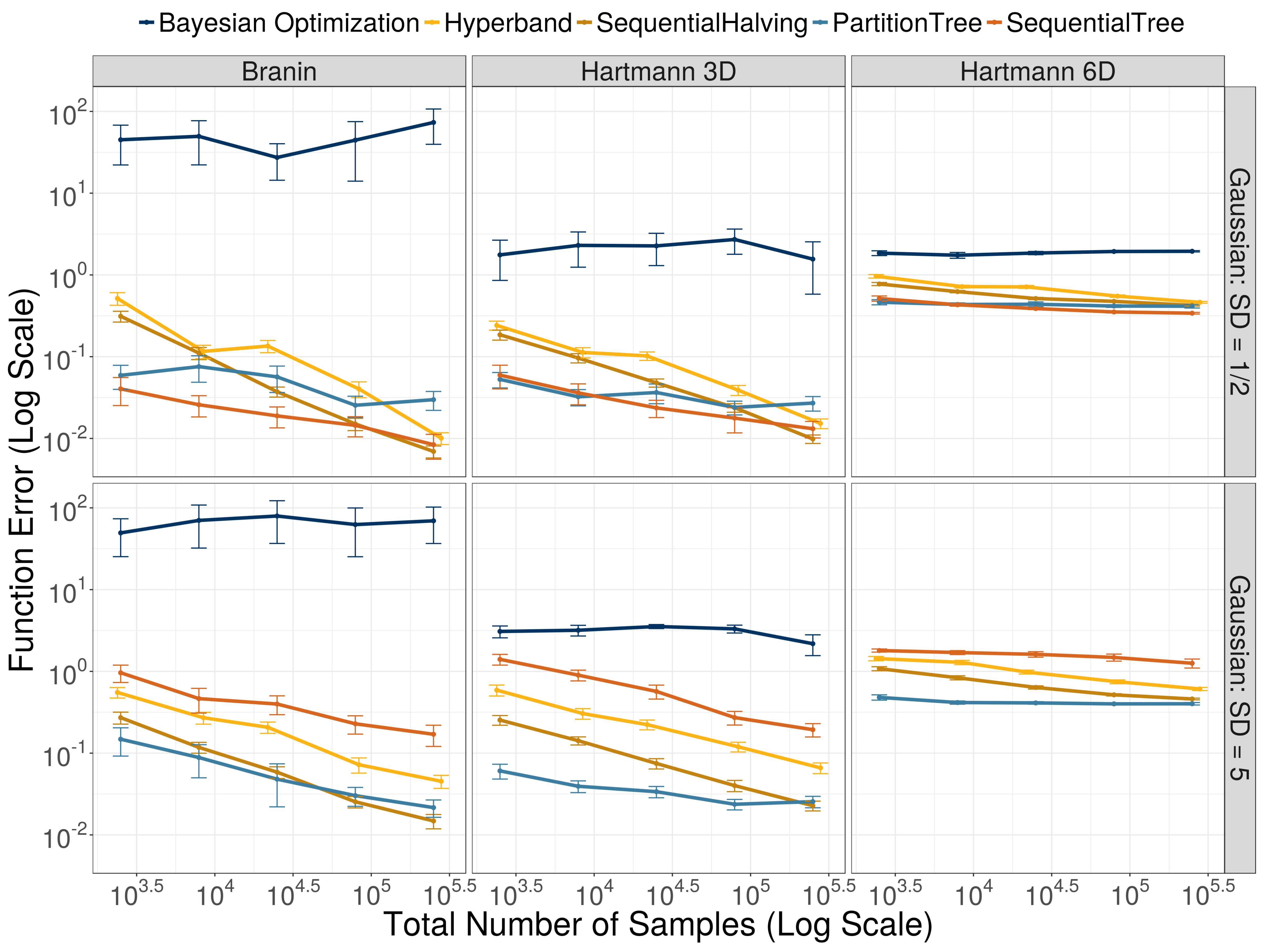
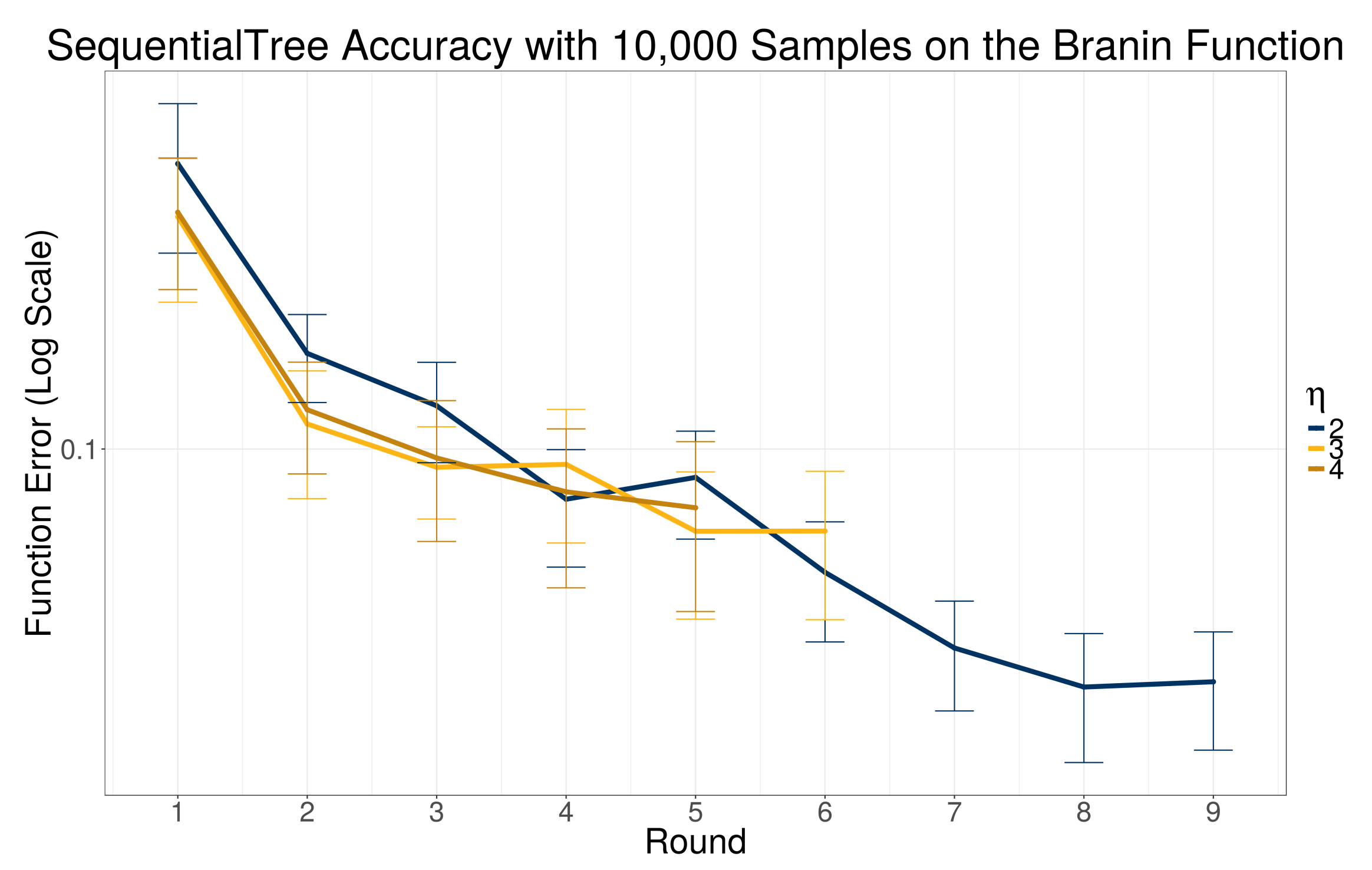
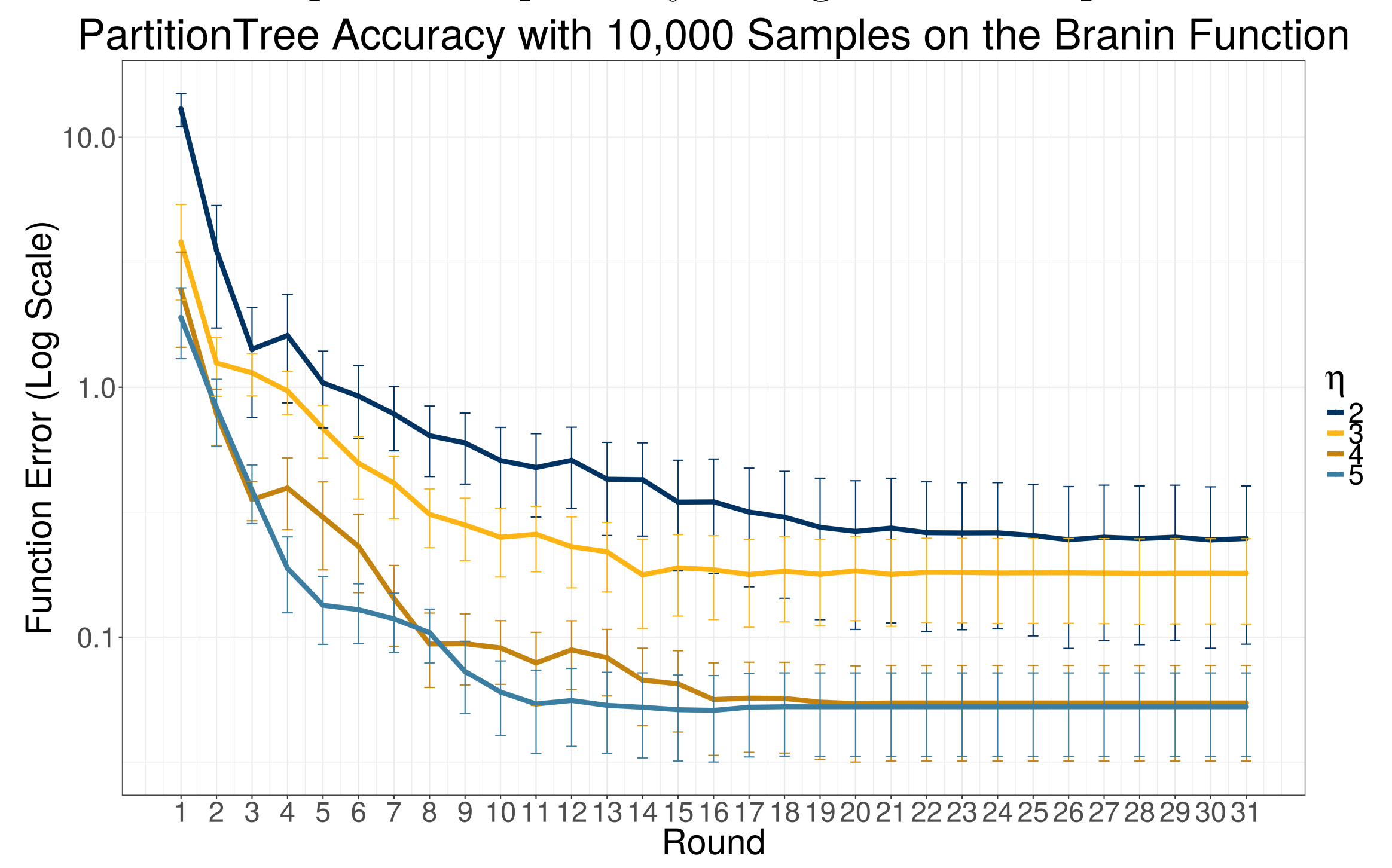


Figure: In the low noise and low budget setting SequentialTree consistently outperforms SequentialHalving and Hyperband. In the high noise setting it does worse, but PartitionTree sometimes performs better than the two bandit algorithms. PartitionTree does not benefit much from more samples, and SequentialTree seems to have a worse rate than the two bandit algorithms.

Algorithm Parameter Settings



(a) With $\eta = 2$ SequentialTree is less aggressive and explores more of the space. Empirically this gives better performance.



(b) For all settings of η, PartitionTree reaches an error floor with successive rounds.

Conclusion and Future Directions

In some low budget settings the tree-based partition algorithms outperform the bandit-based algorithms. For future work we hope to make modifications to the algorithms to achieve faster empirical rates of convergence in terms of sample complexity. We also will explore if any theoretical comparisons can be made.

References

[1] J. Snoek, H. Larochelle, and R. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms.," *NIPS*, 2012.

[2] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *arXiv preprint*, 2016.

[3] Z. Karnin, T. Koren, and O. Somekh, "Almost optimal exploration in multi-armed bandits," *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, vol. 28, pp. 1238–1246, 2013.

[4] K. Jamieson and A. Talwalkar, "Non-stochastic Best Arm Identification and Hyperparameter Optimization," in *Proceedings of AISTATS*, 2015.