The Vizier Gaussian Process Bandit Algorithm

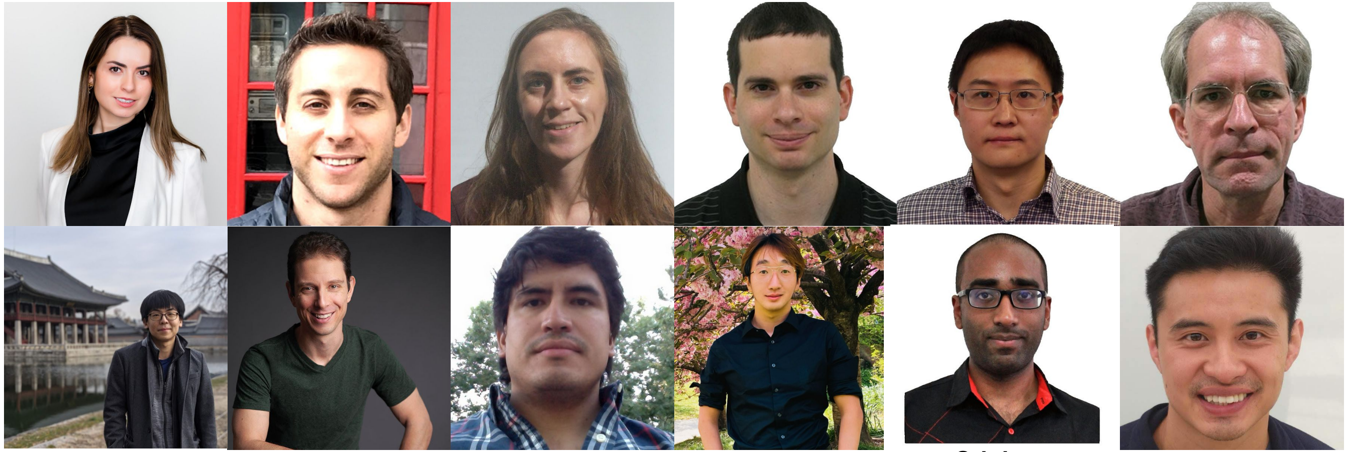
Xingyou (Richard) Song
xingyousong@google.com

On behalf of the Vizier Team

Google DeepMind

Vizier + Extended Team

Setareh Ariafar | Lior Belenki | Emily Fertig | Daniel Golovin | Tzu-Kuo Huang | Greg Kochanski



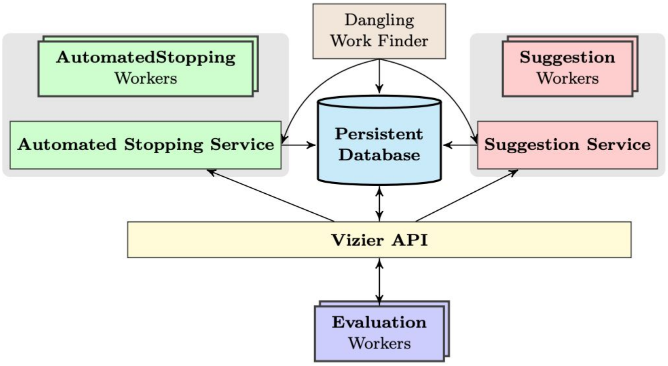
Chansoo Lee | Sagi Perel | Adrian Reyes | Xingyou Song | Srinivas Vasudevan | Richard Zhang

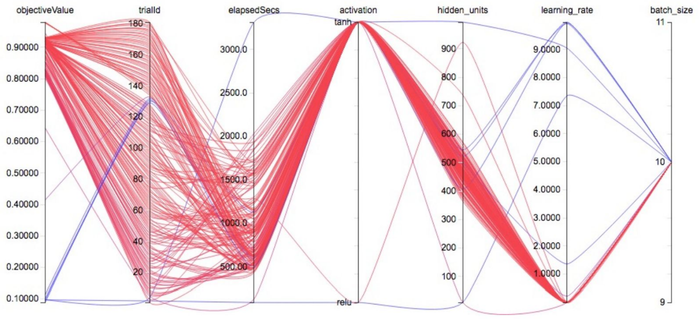
Google DeepMind

Google Vizier (2017)

Google

- Tunes many of Google’s research + products
- O(1K) monthly users, O(70M+) objectives
- Invented before Tensorflow!





Google DeepMind

Notable Users / Downstream Wins

Production

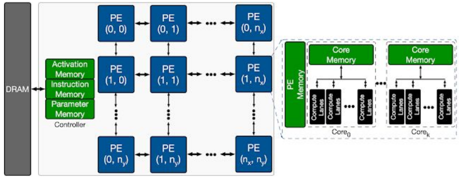
- Search, Ads, Youtube

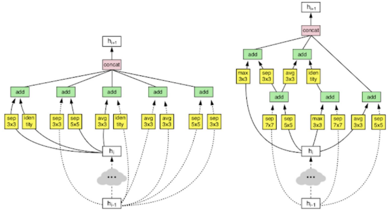
Tuning Research Results:

- Hardware Design, Robotics
- Protein Design

Backend for Evolution:

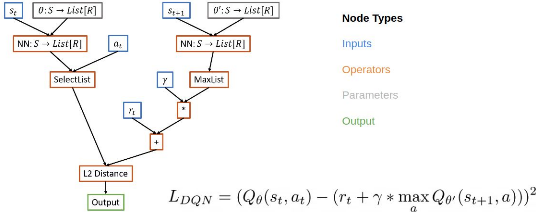
- Neural Architecture Search
- Symbolic Algorithm Search





Normal Cell

Reduction Cell



Google DeepMind

Vertex/Cloud Vizier

- Prod service for external users / businesses
- Shared client API: Easily switch b/w OSS or Cloud

Vertex AI > Documentation > Guides

Was this helpful?

Vertex AI Vizier overview

Send feedback

Vertex AI Vizier is a black-box optimization service that helps you tune hyperparameters in complex machine learning (ML) models. When ML models have many different hyperparameters, it can be difficult and time consuming to tune them manually. Vertex AI Vizier optimizes your model's output by tuning the hyperparameters for you.

Fundamentally Black-box Optimization

- **Optimize unknown function f(x)**
- **Evaluating f(x) is expensive.**
- **Small evaluation budget (~100-100K)**

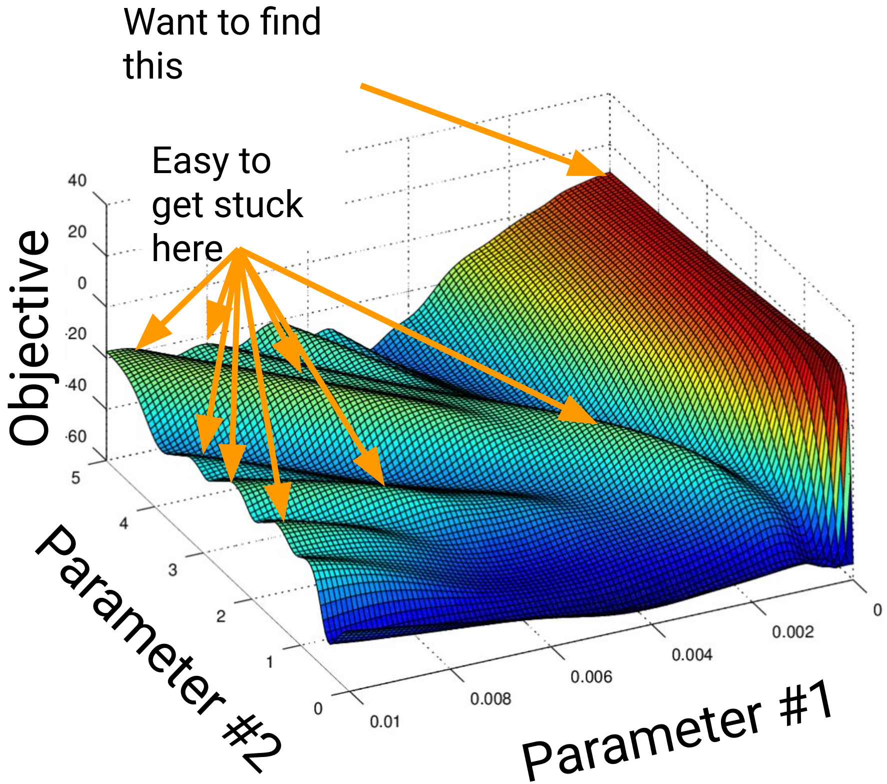


Table of Contents

Service

1. Why a Service?
2. User/Client API: Distributed Tuning

Algorithm

3. Bayesian Optimization Intro
4. Gaussian Process, Acquisitions
5. Batched + Multi-Objective

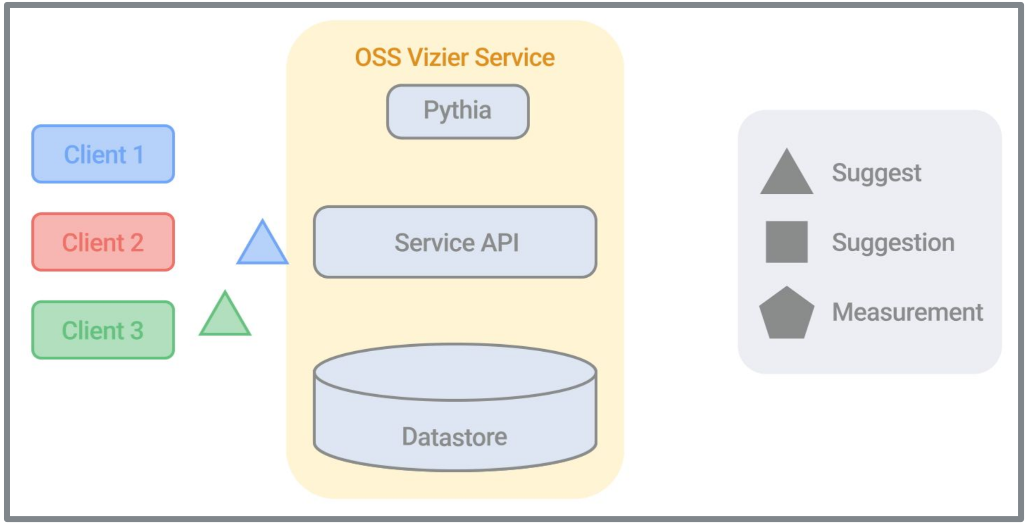
Experiments

6. Baselines + Benchmarks
7. End-to-End Results
8. Ablations

Google DeepMind

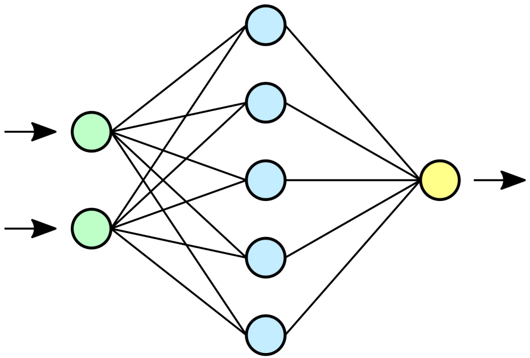
Questions?

Why a Service?

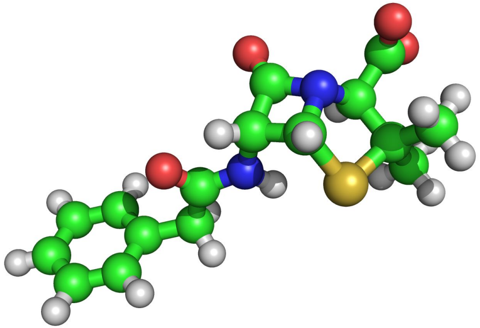


Google DeepMind

The Wide Variety of Scenarios



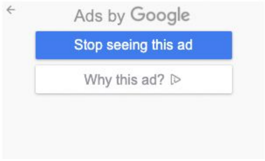
- Tuning large ML model hyperparameters
- [Chemical/Biological processes]
- [Optimizing cookie recipes]



Very different workflows!

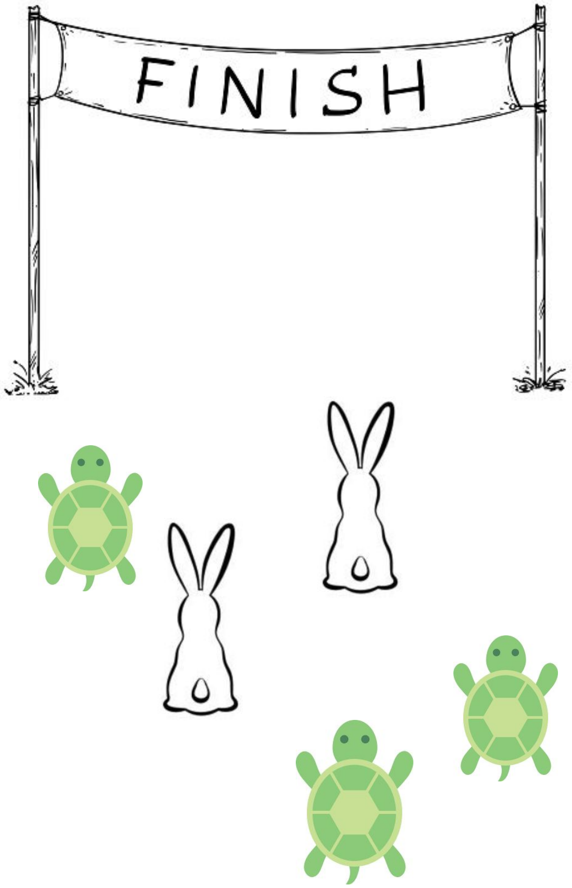
Google's new AI learns by baking tasty machine learning cookies

The system "designs excellent cookies", according to its creators



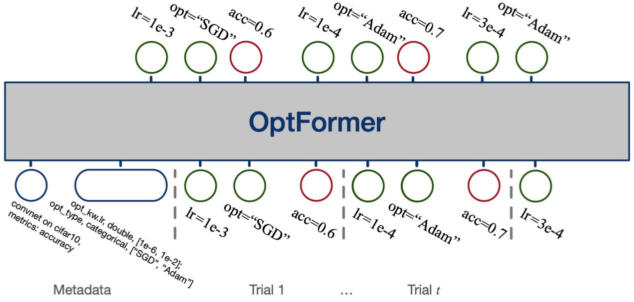
Workflow Possibilities

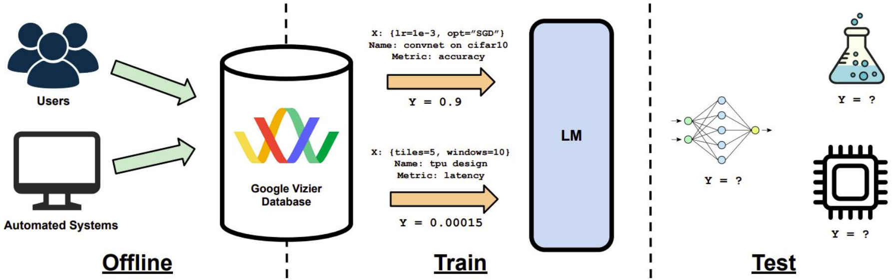
- Eval Latency: Seconds to Weeks
- Eval Budget: 10^1 to 10^7 Trials
- Asynchronous or Synchronous (Batched)
- Failed evaluations: Retried or abandoned
- Early Stopping



Benefits of a Service: No Evaluation Assumptions!

- Users have freedom of when to:
  - Request trials
  - Evaluate Trials
  - Report results
- Service can preserve data on prior usage
  - Led to [OptFormer](), [OmniPred](), and more offline data research!





Setting up Client

```
# Algorithm, search space, and metrics.
study_config = vz.StudyConfig(algorithm='GAUSSIAN_PROCESS_BANDIT')
study_config.search_space.root.add_float_param('w', 0.0, 5.0)
study_config.search_space.root.add_int_param('x', -2, 2)
study_config.search_space.root.add_discrete_param('y', [0.3, 7.2])
study_config.search_space.root.add_categorical_param('z', ['a', 'g', 'k'])
```

```
study = clients.Study.from_study_config(study_config, owner='my_name', study_id='example')
```

Google DeepMind

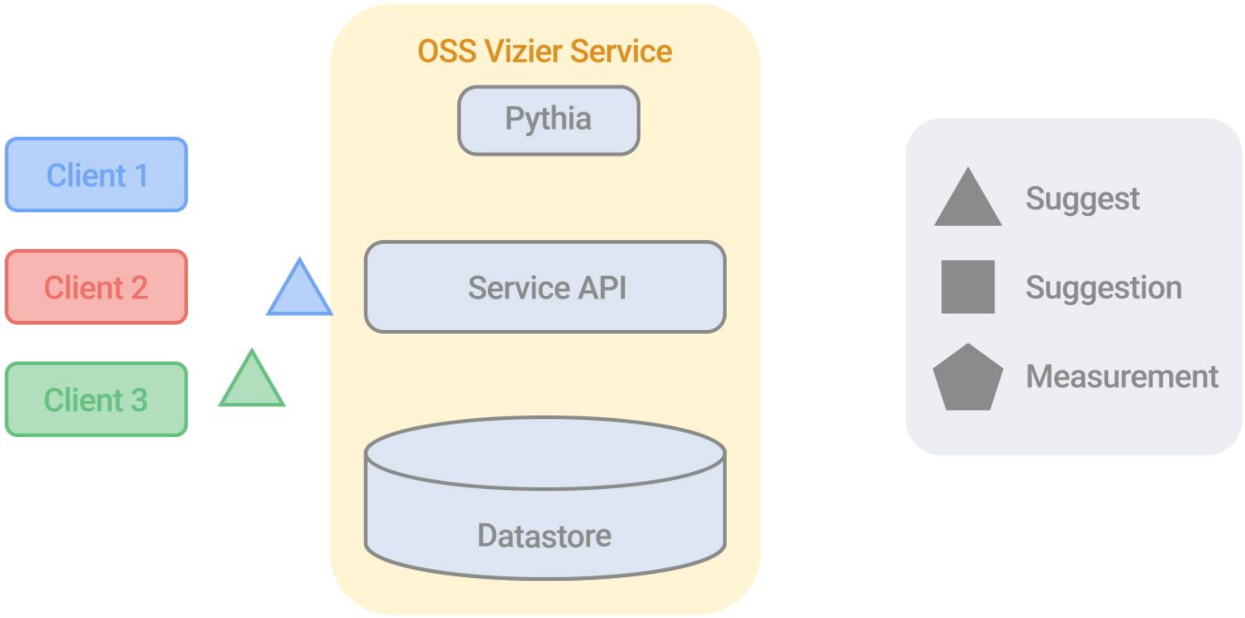
Tuning Loop

Loop involves:

- Client obtains suggestions from server
- Evaluating suggestions
- Completing suggestions + updating server

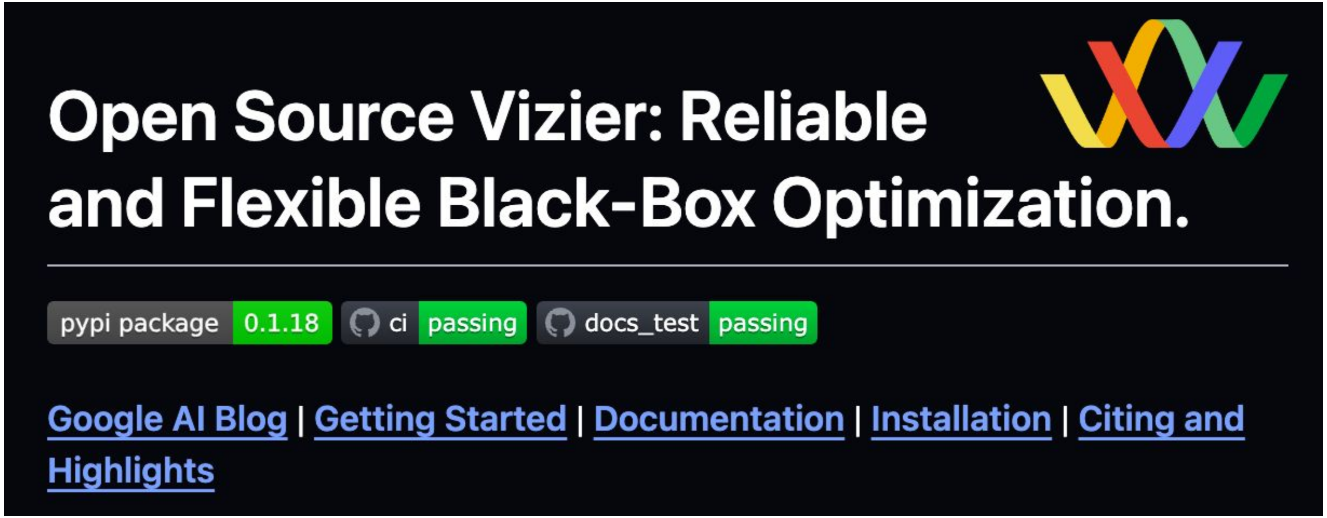
```
for i in range(10):
  suggestions = study.suggest(count=1)
  for suggestion in suggestions:
    params = suggestion.parameters
    objective = evaluate(params['w'], params['x'], params['y'], params['z'])
    suggestion.complete(vz.Measurement({'metric_name': objective}))
```

Suggestion Animation (Full)



OSS Vizier: 2022

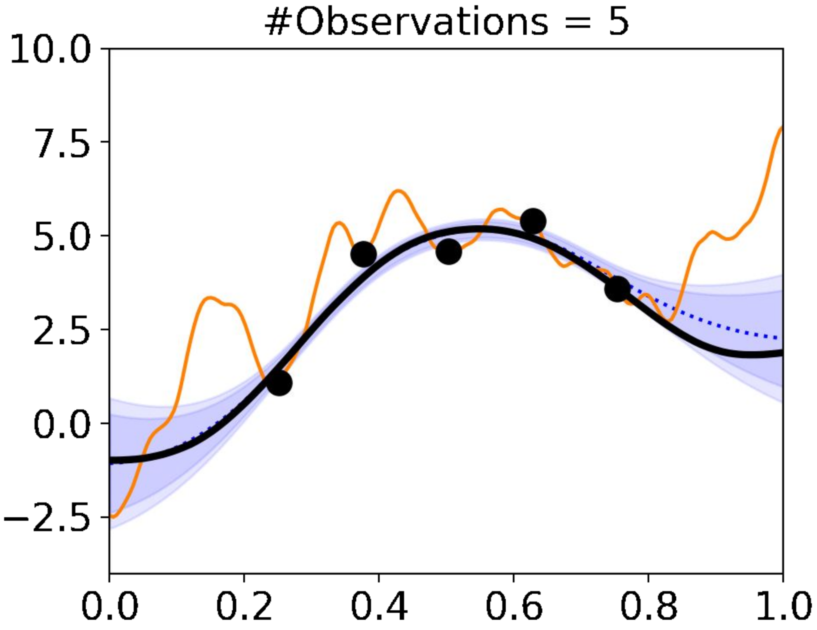
- Standalone + customizable Python codebase
- User can host service



Questions?

Animation from [Wang et al., 2023 "Pre-trained Gaussian processes"]

Algorithm

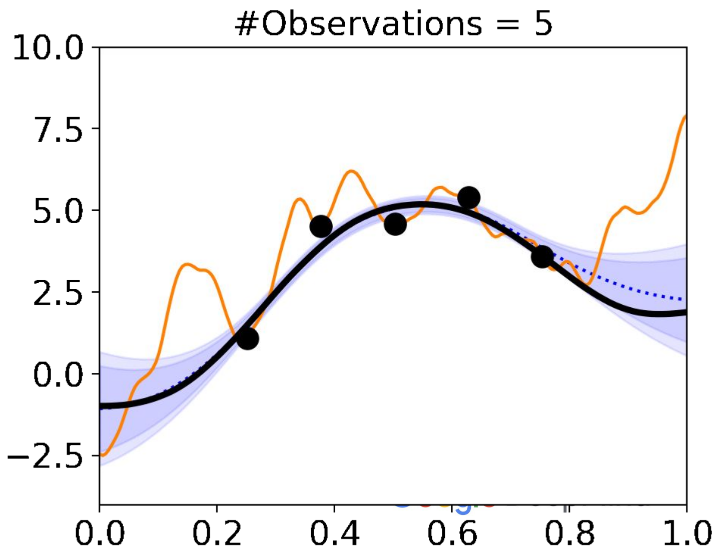


Google DeepMind

Bayesian Optimization Basics

Regressor / Model: Predict outcome y = f(x) from history

- Common Examples
  - Gaussian Process
  - Random Forest
- Guides search: Explore-Exploit
  - Acquisition Function



GP-UCB

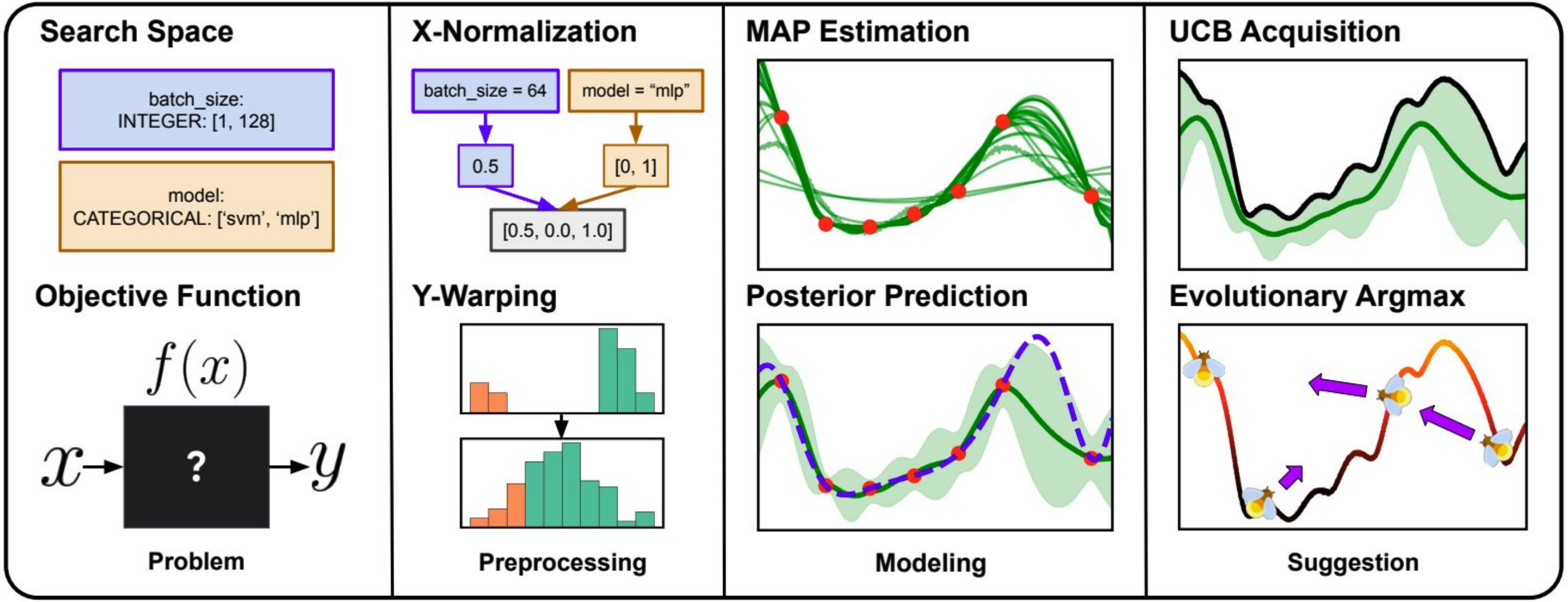
Gaussian Process + Upper Confidence Bound (Srinivas, 2010)

- Provably convergent, sublinear regret

Algorithm 1 Gaussian Process Upper Confidence Bound (GP-UCB) with parameter $\beta > 0$

for $t \in \{1, 2, \ldots\}$ **do**
  Compute posteriors to obtain $\mu_t(x)$ and $\sigma_t(x)$
  $x_t = \arg\max_{x \in \mathcal{X}} \mu_t(x) + \sqrt{\beta} \cdot \sigma_t(x)$
  Evaluate $y_t = f(x_t)$
end for

Key Components



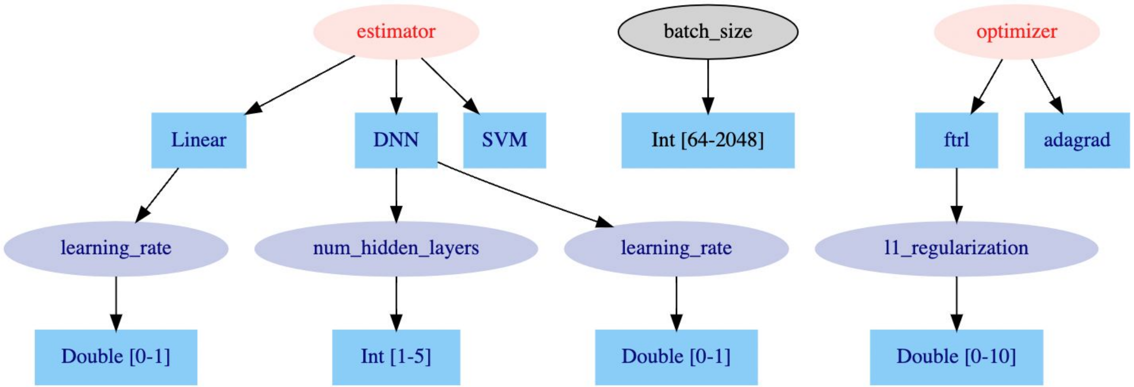
Search Space

Core:

- Double: Continuous range [a,b]
- Integer: Integer range [a,b]
- Discrete: Finite set of floats.
- Categorical: Finite set of strings.

Each ParameterSpec also contains:

- Scaling Type (uniform, log)



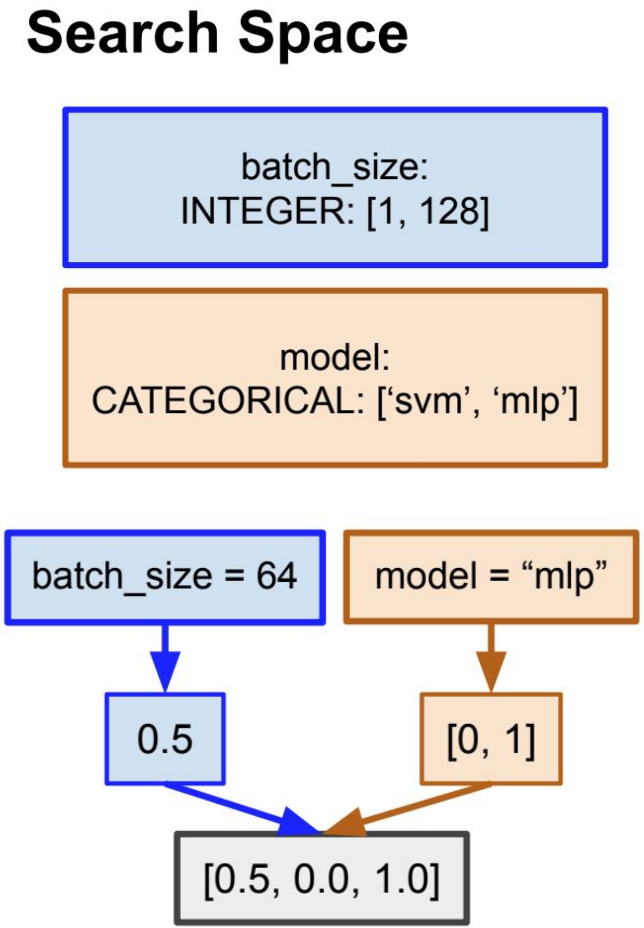
Google DeepMind

X-Normalization

Enforce feature coordinates in [0,1]

- DOUBLE, INTEGER, DISCRETE by min/max bounds
- CATEGORICAL with one-hot

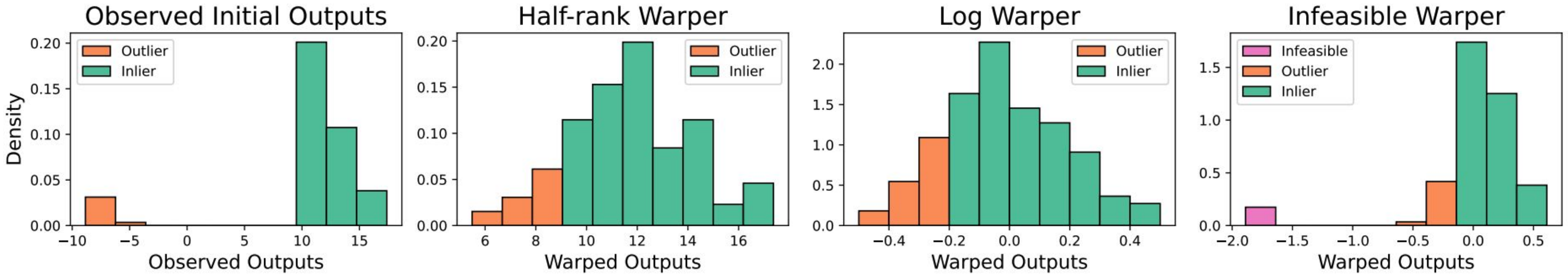
$$\hat{x} \in [0, 1]^D$$



Y-Warping

Raw Y-values can be huge range (e.g. 10^{-7} to 10^{7}), need normalization

- **Half-Rank Warper:** Fit poor y-values to normal curve
- **Log-Warper:** Stretch good-values, compress bad-values
- **Infeasible Warper:** Replace with unpromising values



Gaussian Process Model

Define kernel (distance metric between two points)

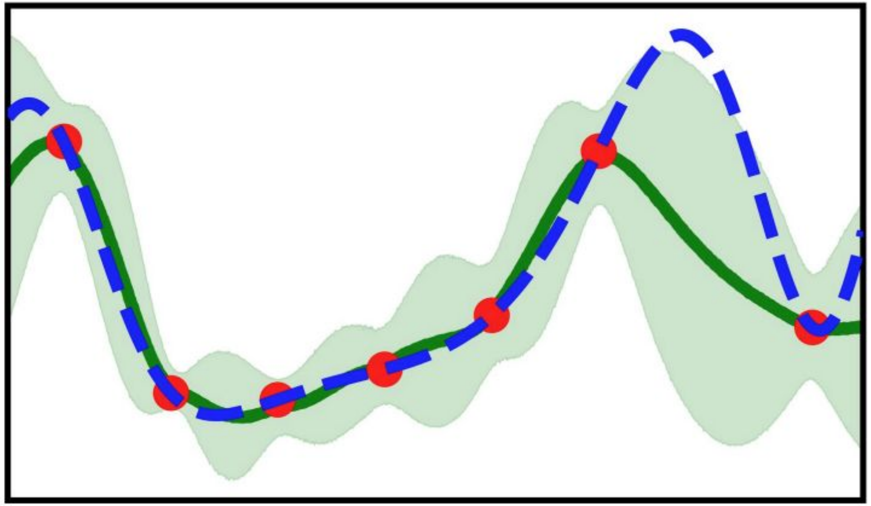
- Matern-5/2 is Euclidean-based

$$\delta(\widehat{x}_i, \widehat{x}_j)^2 = 5 \cdot \sum_{d=1}^{D} \frac{\left(\widehat{x}_i^{(d)} - \widehat{x}_j^{(d)}\right)^2}{\lambda^{(d)}}$$

$$K(\widehat{x}_i, \widehat{x}_j) = \alpha^2 \cdot \left(1 + \delta + \frac{\delta^2}{3}\right) \cdot \exp(-\delta)$$

$$\overline{f} \sim \mathcal{GP}(0, K)$$

$$\widehat{y} \sim \mathcal{N}\left(\overline{f}(\widehat{x}), \exp(\varepsilon_{\log})\right)$$



MAP Estimation

Fit GP hyperparameters to maximize likelihood: $\Pr(\{\widehat{x}_s, \widehat{y}_s\}_{s=1}^{t}, \alpha_{\log}, \overrightarrow{\lambda}_{\log}, \varepsilon_{\log})$

- L-BFGS-B optimizer

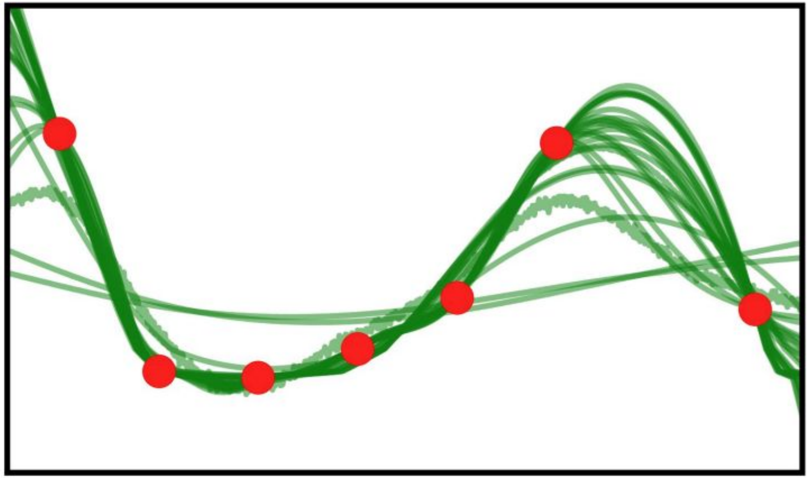
$$\log \Pr(\alpha_{\log}) + \log \Pr(\overrightarrow{\lambda}_{\log}) + \log \Pr(\varepsilon_{\log}) + \log \Pr(\{\widehat{x}_s, \widehat{y}_s\}_{s=1}^{t}; \alpha_{\log}, \overrightarrow{\lambda}_{\log}, \varepsilon_{\log})$$

<u>GP Hyperparameter Distribution</u>

$$\alpha_{\log} \sim \mathcal{N}_{[-3,1]}(\log 0.039, 50)$$

$$\lambda_{\log}^{(d)} \sim \mathcal{N}_{[-2,1]}(\log 0.5, 50)$$

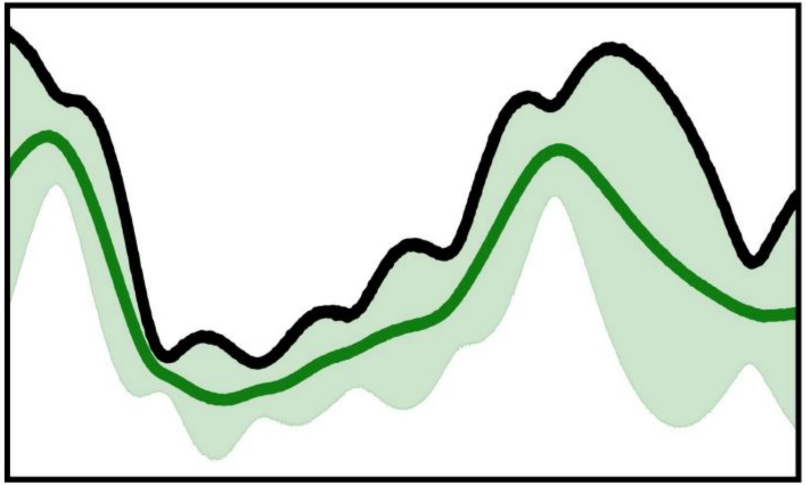
$$\varepsilon_{\log} \sim \mathcal{N}_{[-10,0]}(\log 0.0039, 50)$$



UCB Acquisition

- Mean + standard deviation
- Explicit exploration-exploitation tradeoff

$$\mathrm{UCB}(x) = \mu_t(x) + \sqrt{\beta} \cdot \sigma_t(x)$$



Acquisition: Trust Region

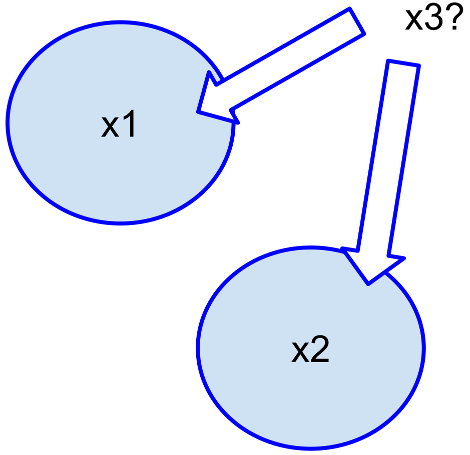
UCB Coefficient is high: $\sqrt{\beta} = 1.8$

UCB(x) is highest around corners.

$$\mathrm{UCB}(x) = \mu_t(x) + \sqrt{\beta} \cdot \sigma_t(x)$$

Use trust region around previous "trusted" points!

$$x \mapsto \begin{cases} \mathrm{UCB}(x) & \text{if } \mathrm{dist}(\widehat{x}, \text{trusted}) \le \text{radius} \\ -10^{12} - \mathrm{dist}(\widehat{x}, \text{trusted}) & \text{if } \mathrm{dist}(\widehat{x}, \text{trusted}) > \text{radius} \end{cases}$$



Google DeepMind

Acquisition Function Optimization

Before AutoGrad: **Everything in C++.**

- Multithreaded, not GPU-accelerated

Zeroth-Order Optimization (No gradients!)

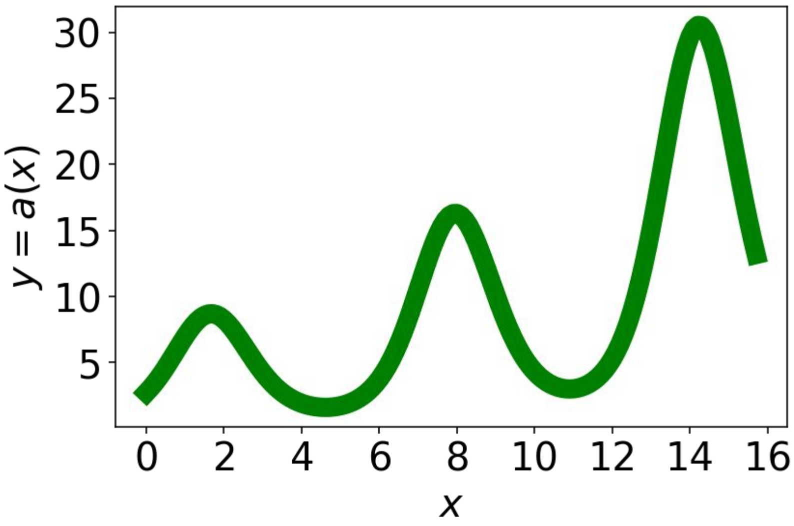
- “Hopping up a hill”

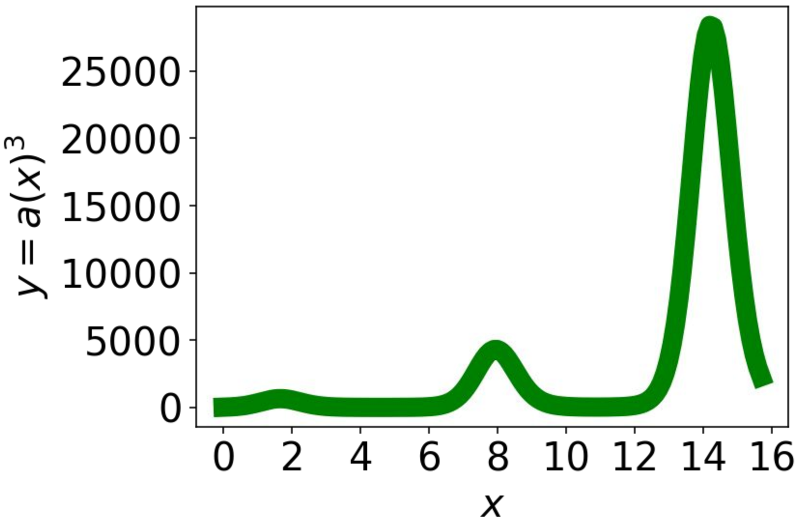
Google DeepMind

AF Optimization: Zeroth-Order Benefits

"Is a(x) > a(x')?"

- **Invariant to function scale.**



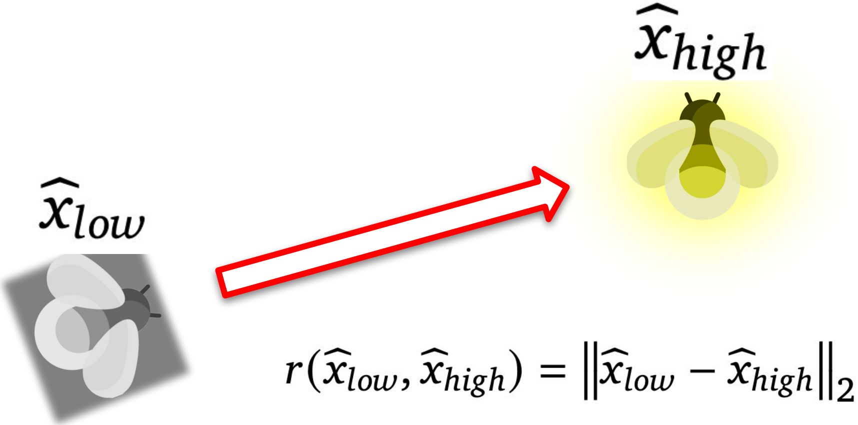


Google DeepMind

AF Optimization: Firefly Algorithm

Pick two random “fireflies” and move “dimmer” towards “brighter” based on distance

$$\widehat{x}_{low} \leftarrow \widehat{x}_{low} + \eta \exp(-\gamma \cdot r^2) \cdot (\widehat{x}_{high} - \widehat{x}_{low}) + \mathcal{N}(0, \omega^2 I_D)$$

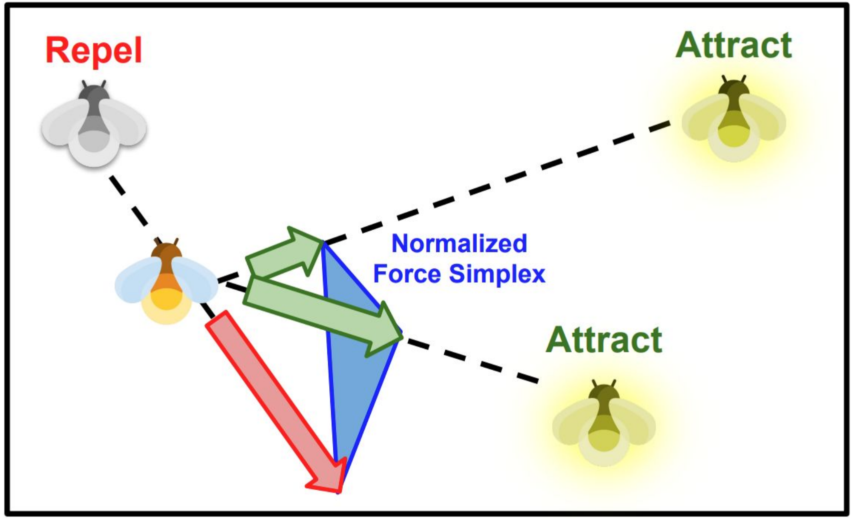


$$r(\widehat{x}_{low}, \widehat{x}_{high}) = \left\|\widehat{x}_{low} - \widehat{x}_{high}\right\|_2$$

AF Optimization: Vectorized Firefly

Write Algorithm in JAX for crucial speedups!

- Vectorization (simultaneous forces from other fireflies)
- Support CATEGORICALs
- Repel from dimmer fireflies



Questions?

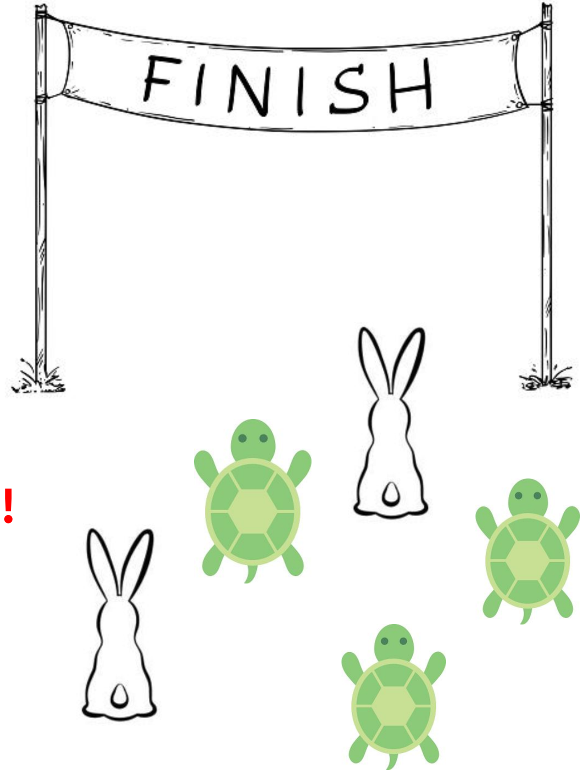
Batched Optimization: Motivation

Users request:

- Batch of X’s
- More X’s w/o evaluating previous

Naively sample from sequential algorithm:

- **Argmax(UCB(history)) leads to duplicate trials!**



Batched Optimization: Pure Exploration

1. Give dummy values (0) to unevaluated objectives.

$$\mathcal{U}_t = \{x_v, 0\}_{v=1}$$

2. Maximize standard deviation over real + "dummy" history:

$$\sigma_t(x|\mathcal{D}_t \cup \mathcal{U}_t)$$

3. Make sure UCB is still better than historical maximum

$$\sigma_t(x|\mathcal{D}_t \cup \mathcal{U}_t) + \boxed{\begin{array}{c}\rho \cdot \min\left(\mathrm{UCB}(x|\mathcal{D}_t, \emptyset, \beta_e) - \tau_t, 0\right) \\ \tau_t \; := \; \mu_t(x_t^*|\mathcal{D}_t)\end{array}}$$

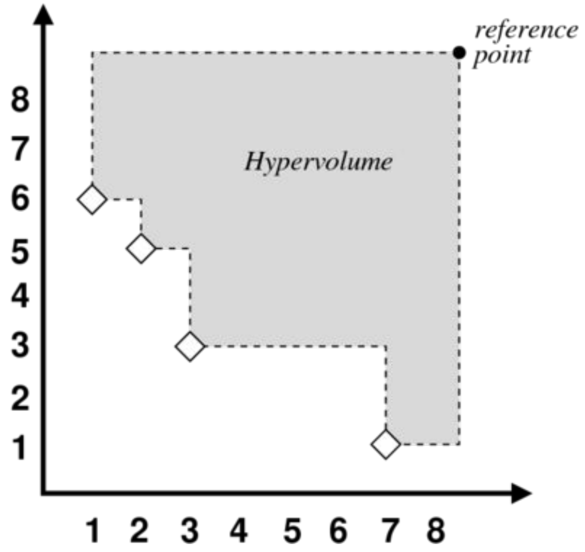
Google DeepMind

Multi-Objective Optimization

M metrics, find points which maximize hypervolume

- volume of Pareto frontier
- w.r.t. reference point

Exact computation given points is #P-Hard!



Scalarization: Intuition

Suppose we want to maximize:

$$f(x) = (f^{(1)}(x), f^{(2)}(x))$$

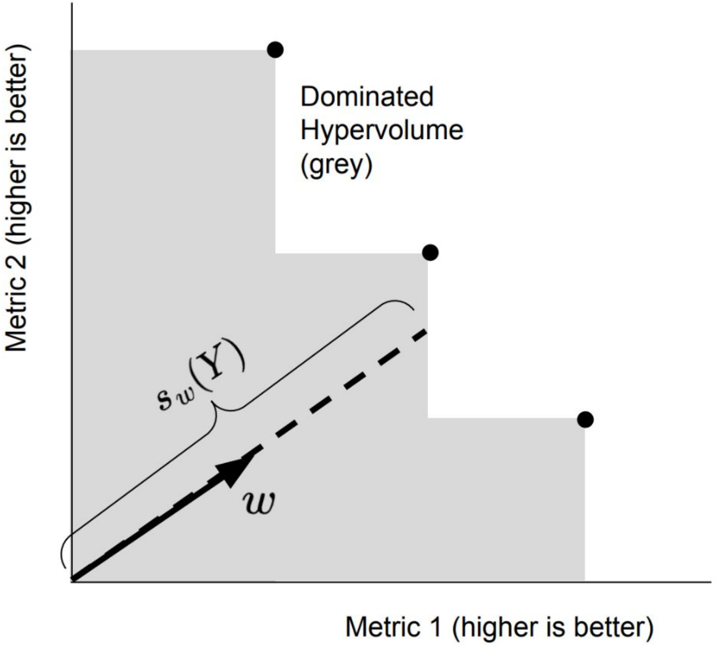
Try weighted sum, ex:

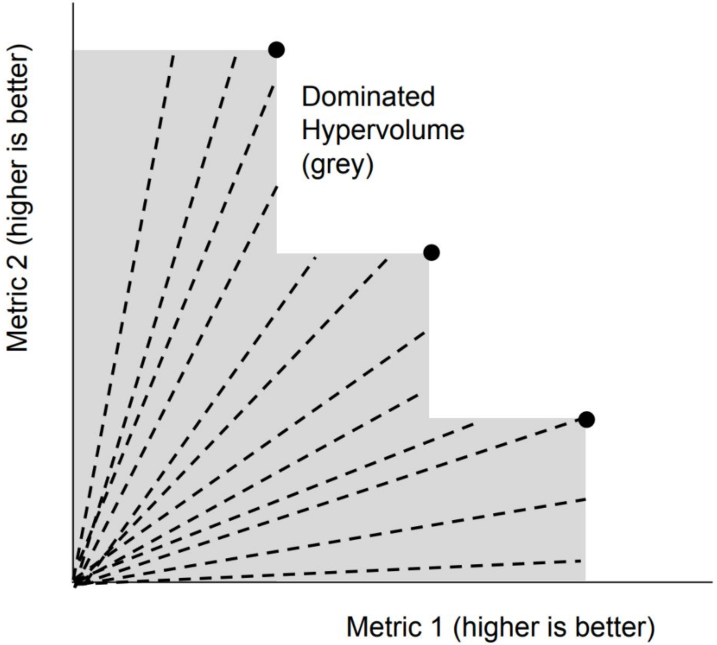
$$s_{(0.4,0.6)}(f(x)) = 0.4 \cdot f^{(1)}(x) + 0.6 \cdot f^{(2)}(x)$$

Google DeepMind

Hypervolume Integral

Hypervolume is integral over polar coordinates!

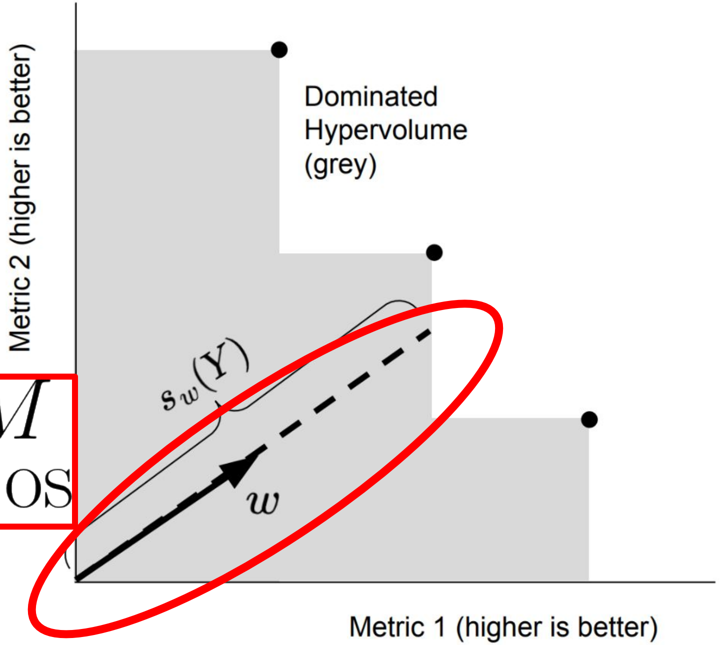


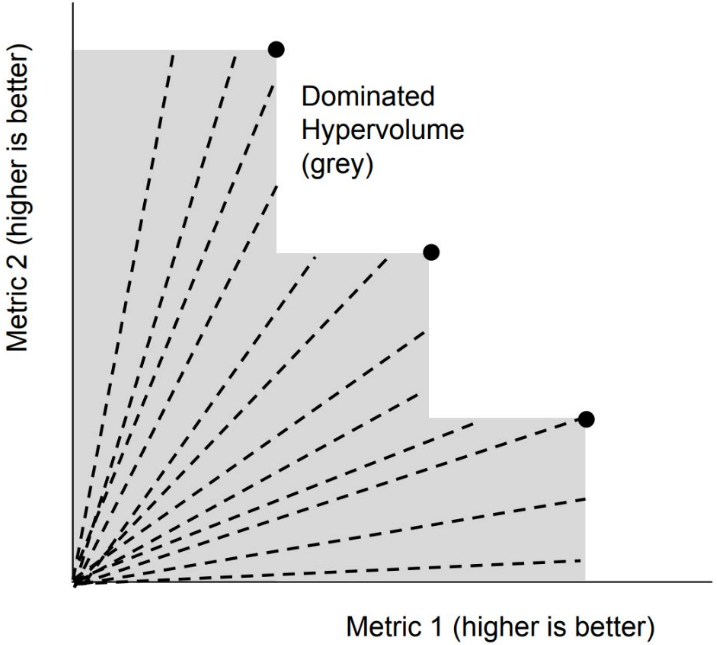


Hypervolume Integral

Hypervolume is integral over polar coordinates!

$$w \sim S_{\text{pos}}^{M}$$

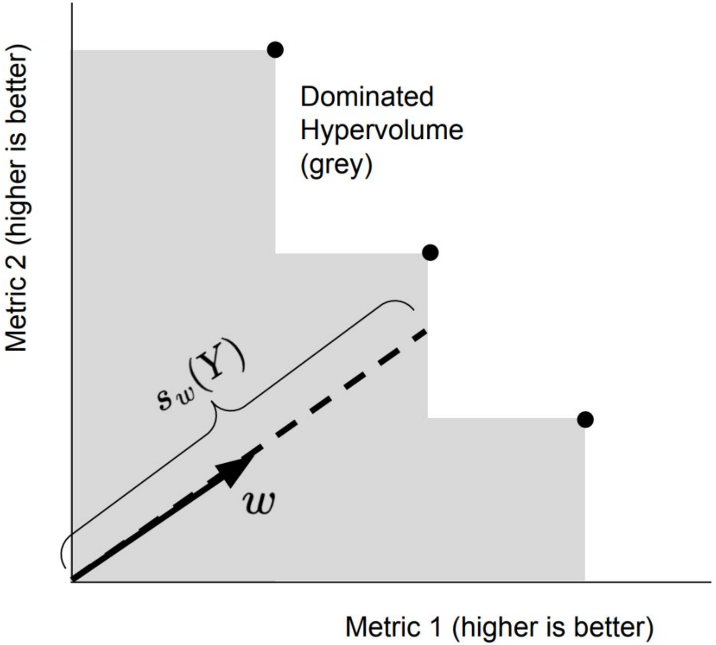


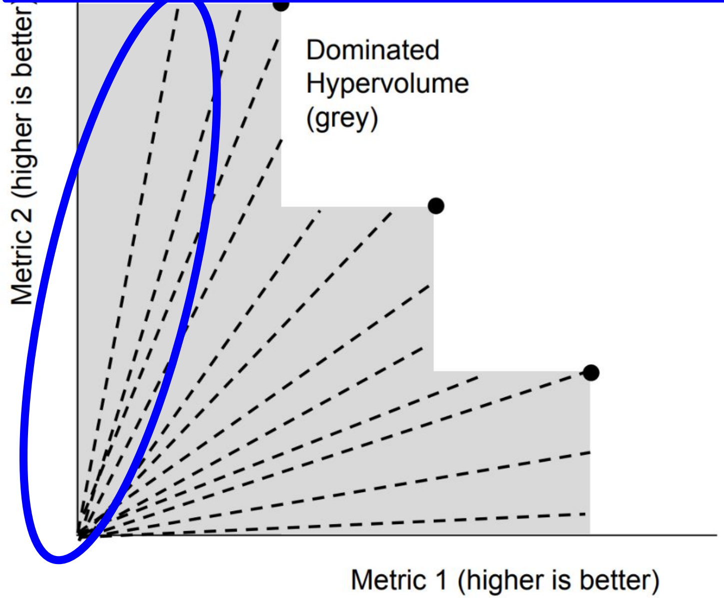


Hypervolume Integral

Hypervolume is integral over polar coordinates!

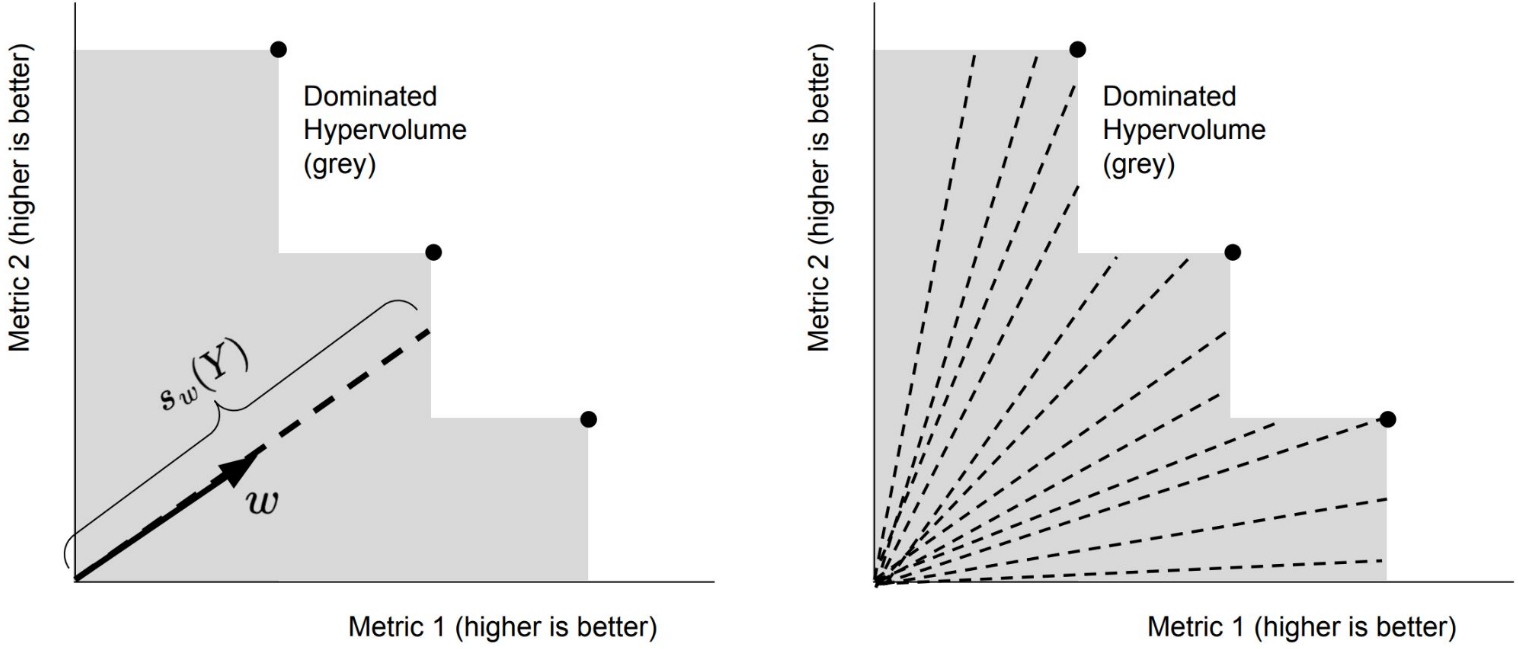
$$\boxed{s_w(y) = \left(\min_{1 \le m \le M} \mathrm{ReLU}(y^{(m)}/w^{(m)}) \right)^M}$$





Hypervolume Integral

$$\boxed{HV(\{y_1, \ldots, y_k\}) \approx \mathbb{E}_w[s_w(y - y_{ref})]}$$



UCB + Hypervolume

Compute UCB's over all metrics:

$$\overrightarrow{\mathrm{UCB}}(x) = (\mathrm{UCB}^{(1)}(x), \ldots, \mathrm{UCB}^{(M)}(x))$$

Scalarize Acquisition:

$$\mathbb{E}_w \left[s_w(\overrightarrow{\mathrm{UCB}}(x)) \right]$$

Hypervolume Improvement:

$$\mathbb{E}_w \left[\max\{\max_{y \in \mathcal{D}_t}\{s_w(y)\}, s_w(\overrightarrow{\mathrm{UCB}}(x))\} \right]$$

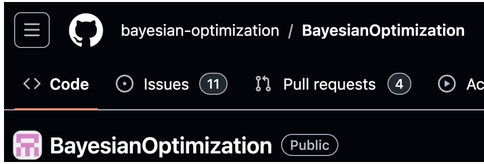
Google DeepMind

Questions?

Experiments

Algorithmic Baselines

- Ax / BoTorch (Meta)
- BayesianOptimization
- HEBO
- HyperOpt
- Optuna
- Scikit-Optimize



~50-90% of all use-cases!

Method Similarities

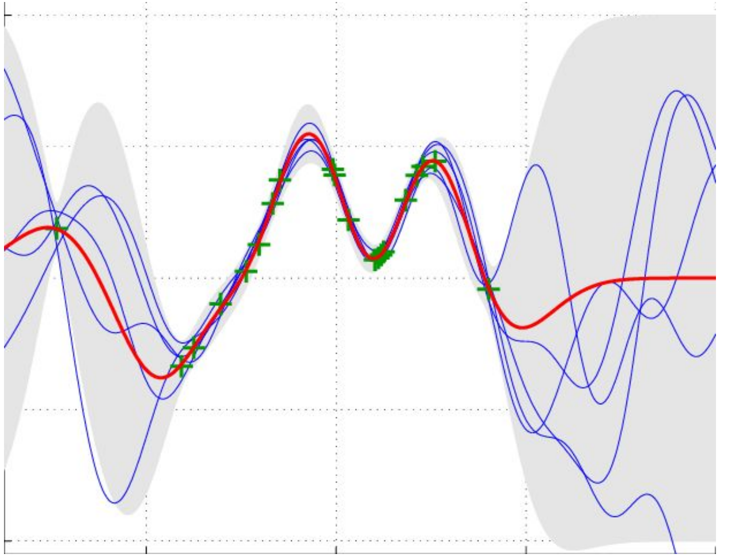
TPE-based: HyperOpt, Optuna

GP-based: Ax, BayesianOptimization, HEBO, SkOpt

- Matern-5/2 Kernel (Continuous)
- L-BFGS-B MAP Estimation

$$\delta(\widehat{x}_i, \widehat{x}_j)^2 = 5 \cdot \sum_{d=1}^{D} \frac{\left(\widehat{x}_i^{(d)} - \widehat{x}_j^{(d)}\right)^2}{\lambda^{(d)}}$$

$$K(\widehat{x}_i, \widehat{x}_j) = \alpha^2 \cdot \left(1 + \delta + \frac{\delta^2}{3}\right) \cdot \exp\left(-\delta\right)$$



GP Method Differences

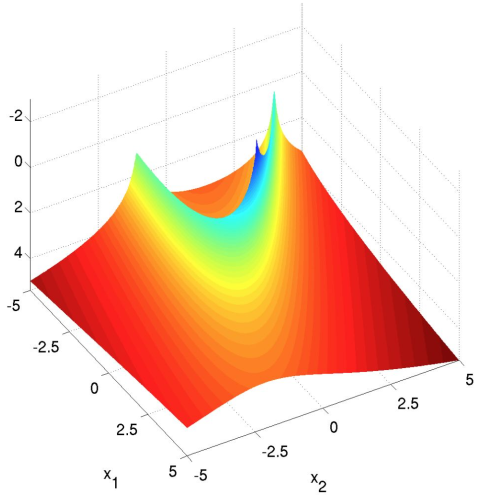
Algorithm	Acquisition Function	Acquisition Optimizer
Ax	Expected Improvement (EI)	L-BFGS-B (Continuous) Hill-Climb (Mixed/Categorical)
Bayesian-Optimization	UCB (Coeff = 2.5)	L-BFGS-B
HEBO	Max(EI, PI, UCB)	NSGA2
SkOpt	Random(EI, LCB, PI)	L-BFGS-B (Continuous + Mixed) Random Search (Categorical)
Vizier (ours)	UCB (Coeff = 1.8)	Firefly

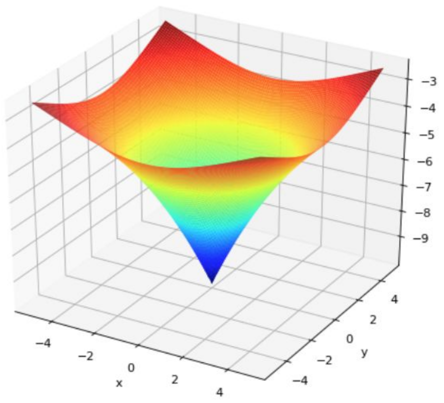
Google DeepMind

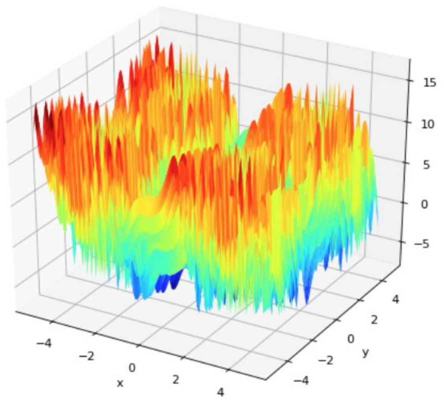
Benchmark Functions

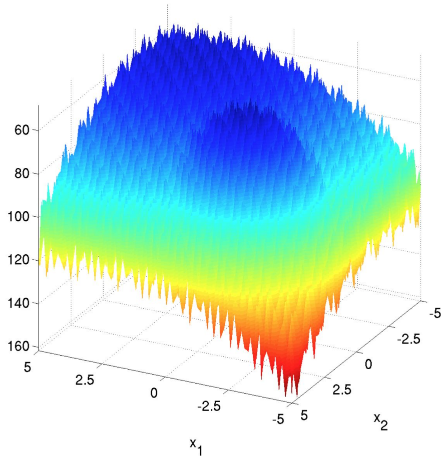
Emphasis on different shapes

- Blackbox Optimization Benchmark (BBOB)
- COMBO
- Multi-Objective (DTLZ, WFG, ZDT)



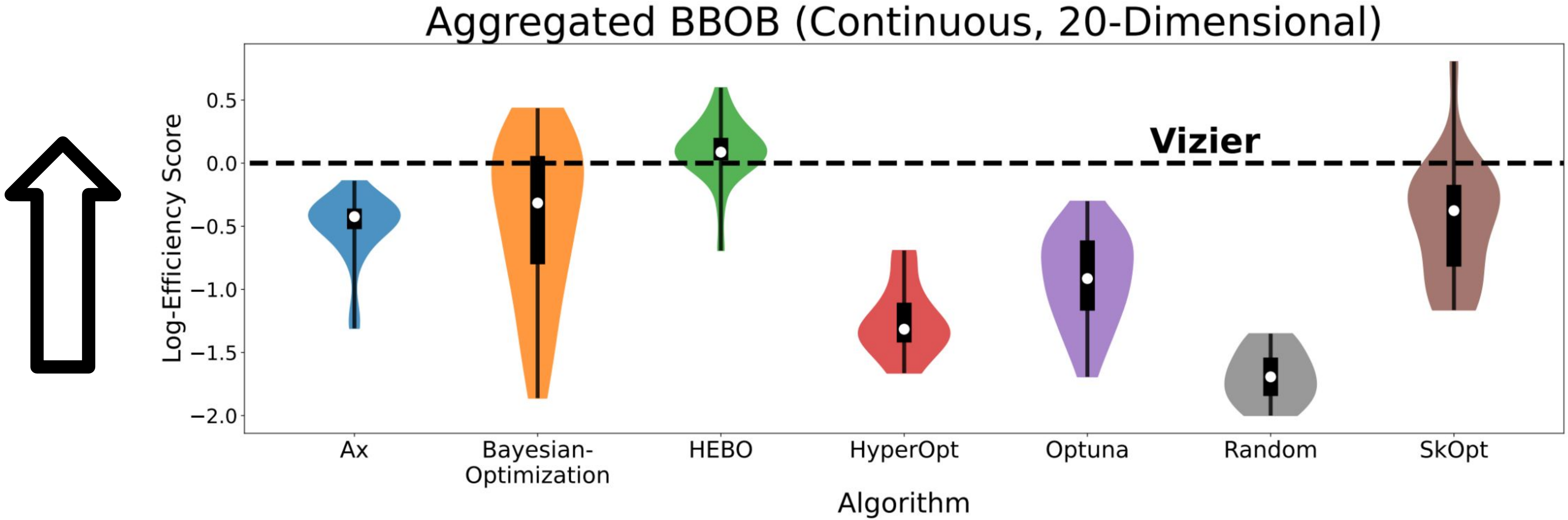




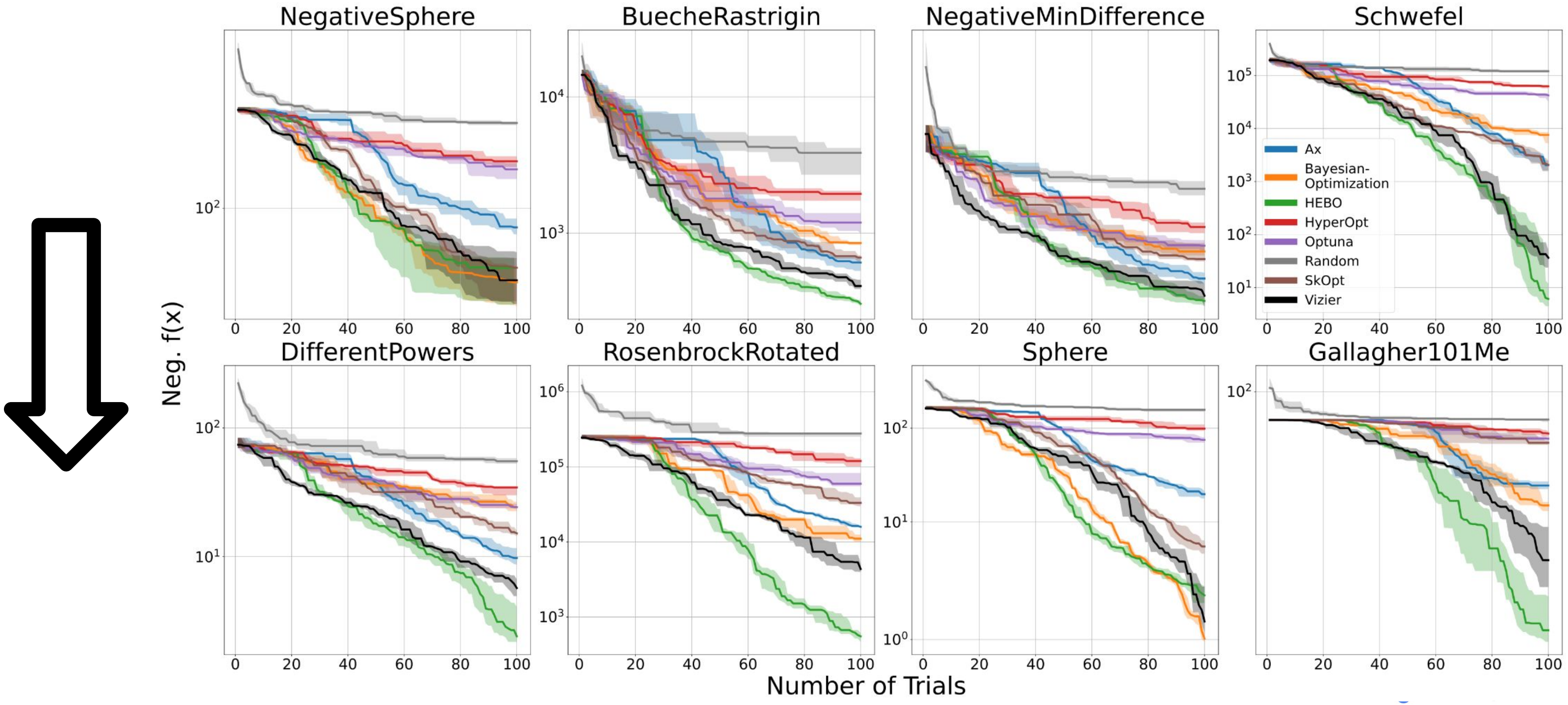


Most Common: Full Continuous (20D)

$$\mathrm{LogEfficiency}(y \mid f, \mathcal{A}) = \log\left(\frac{\mathrm{RequiredBudget}(y \mid f, \mathrm{Vizier})}{\mathrm{RequiredBudget}(y \mid f, \mathcal{A})}\right)$$

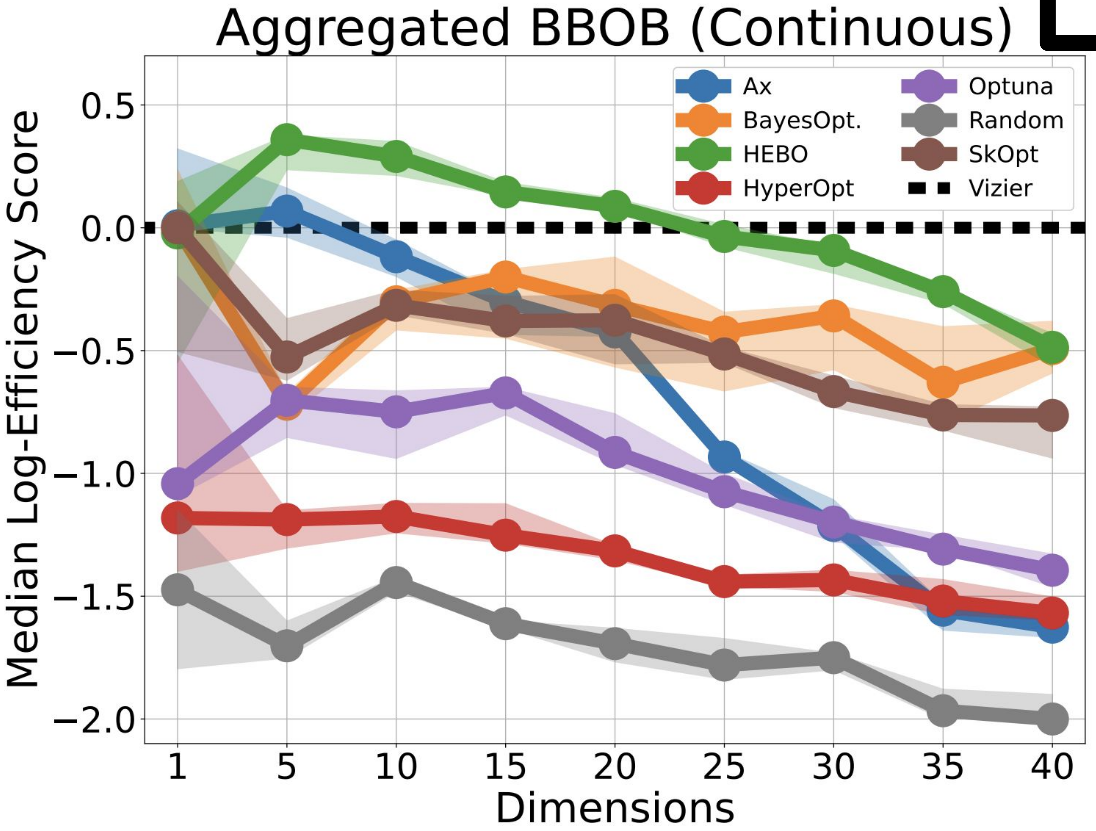


Full Continuous (20D) Individual Plots



Continuous (Varied Dimension)

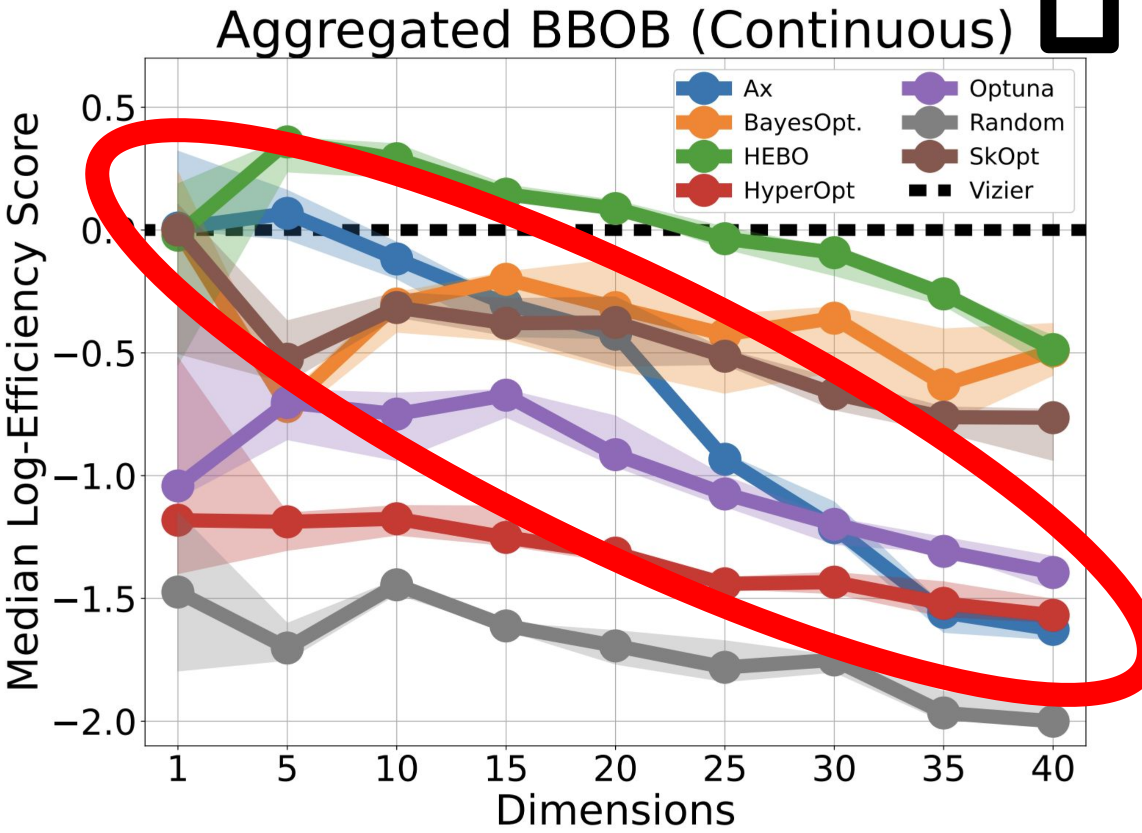
Giant “kitchen-sink” spaces



Continuous (Varied Dimension)

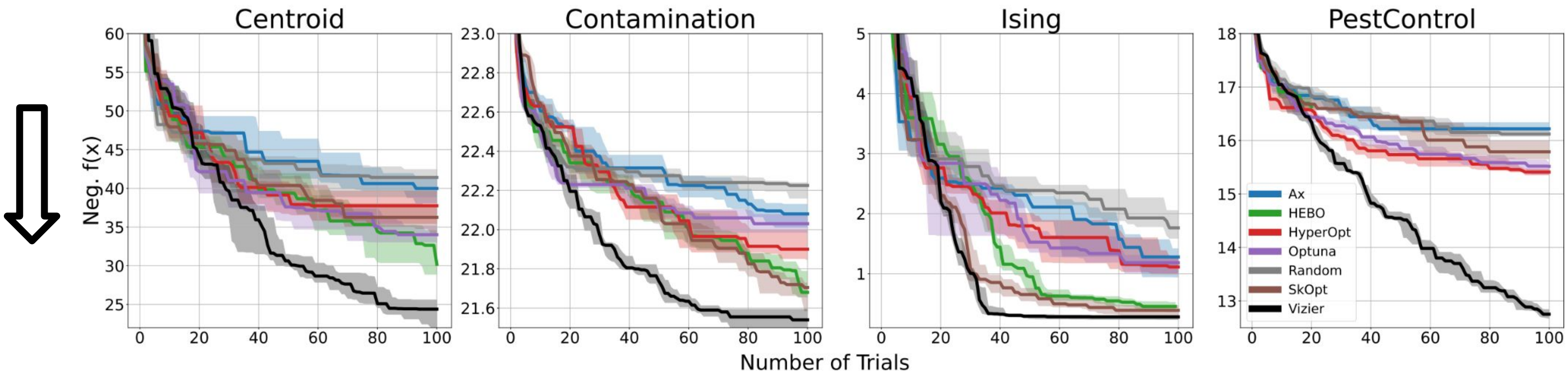
Giant “kitchen-sink” spaces

High-Dim Degradation



Pure Categorical

Vizier > Everyone Else significantly.

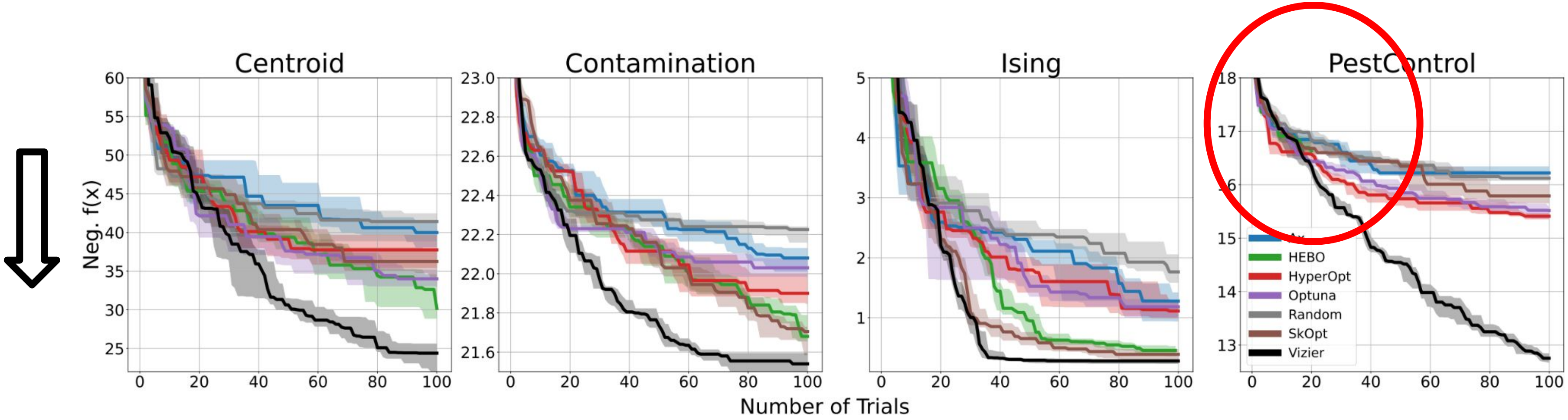


Google DeepMind

Pure Categorical

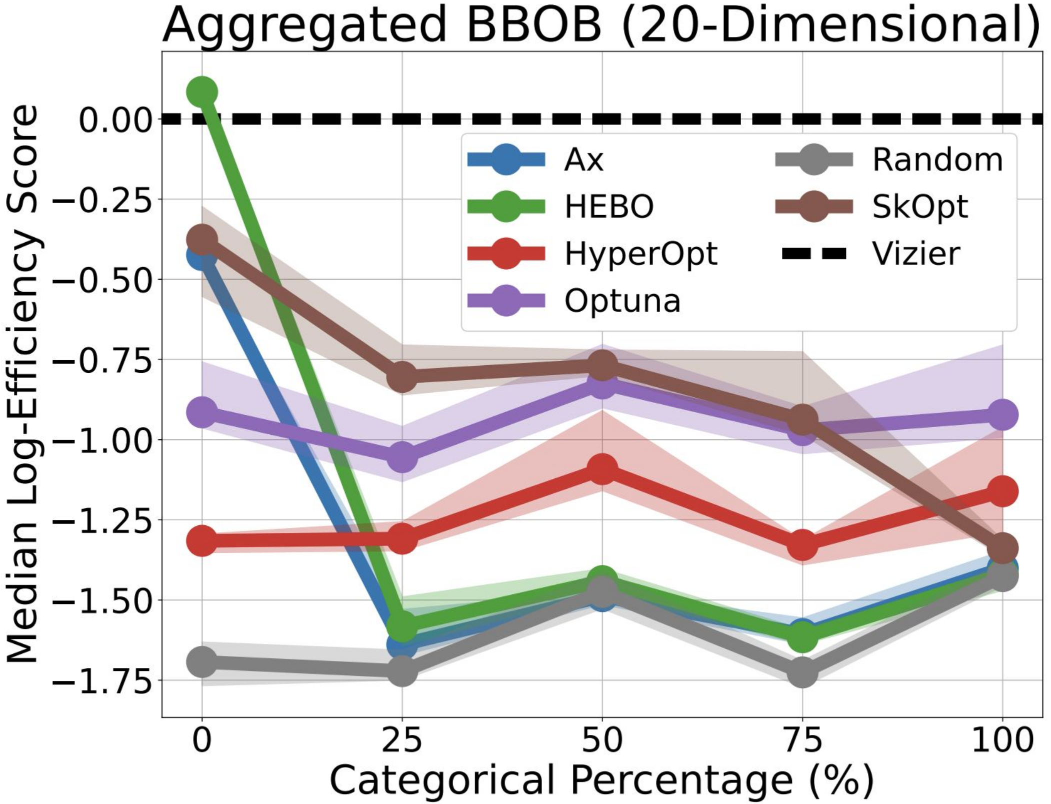
Vizier > Everyone Else significantly.

HEBO crashes on non-boolean spaces.



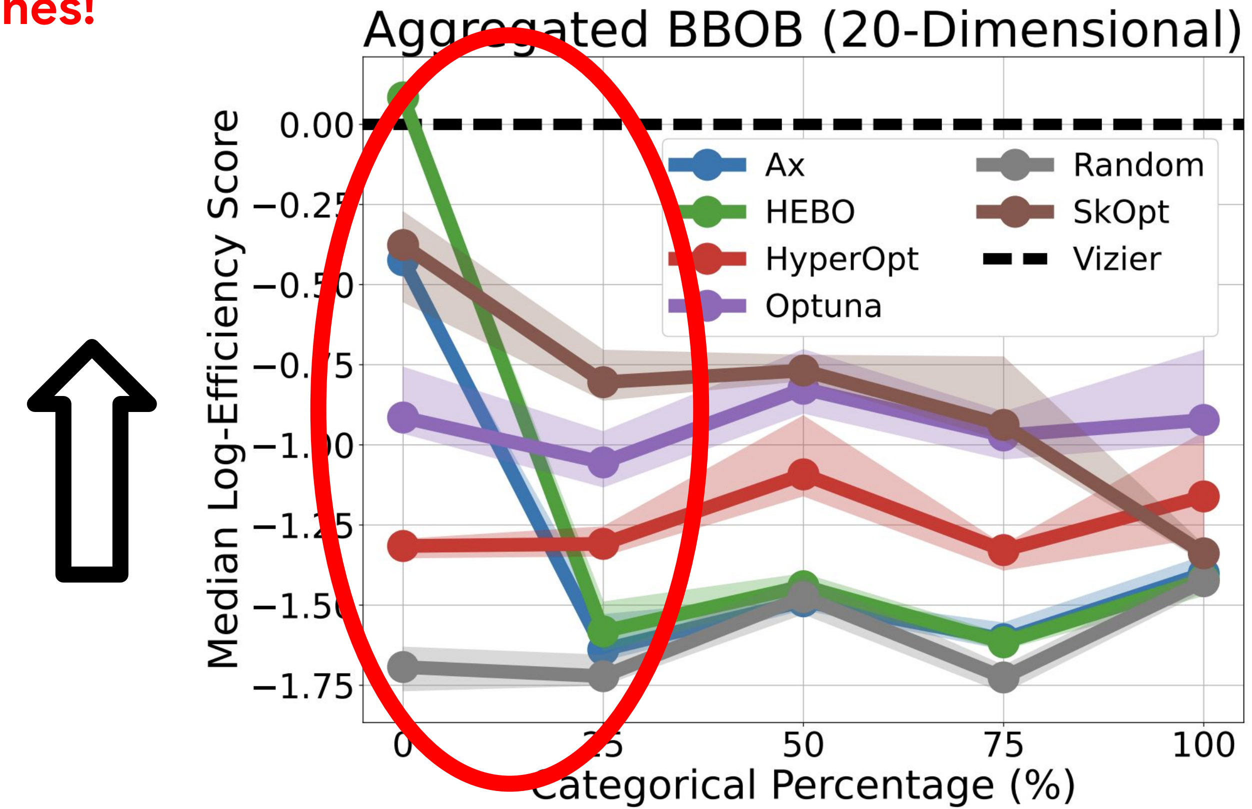
Google DeepMind

Hybrid (Continuous + Categorical)

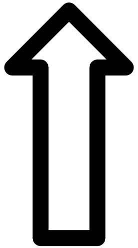


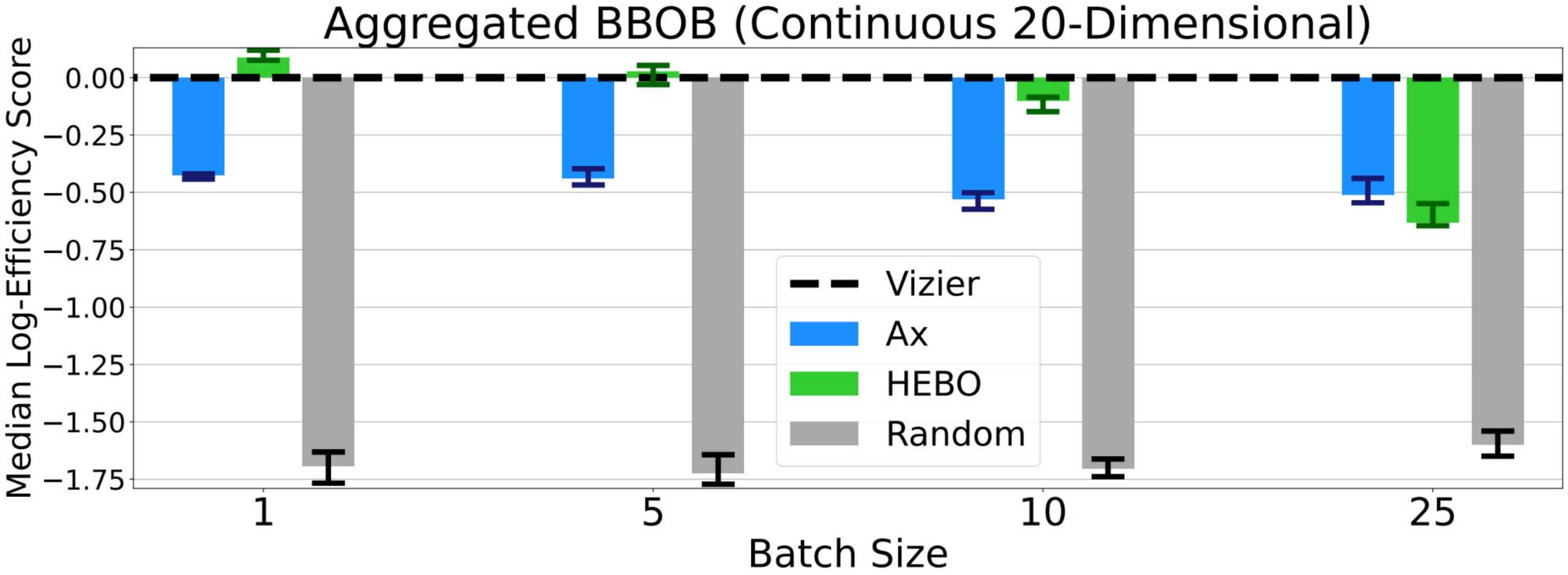
Hybrid (Continuous + Categorical)

Massive drop in baselines!



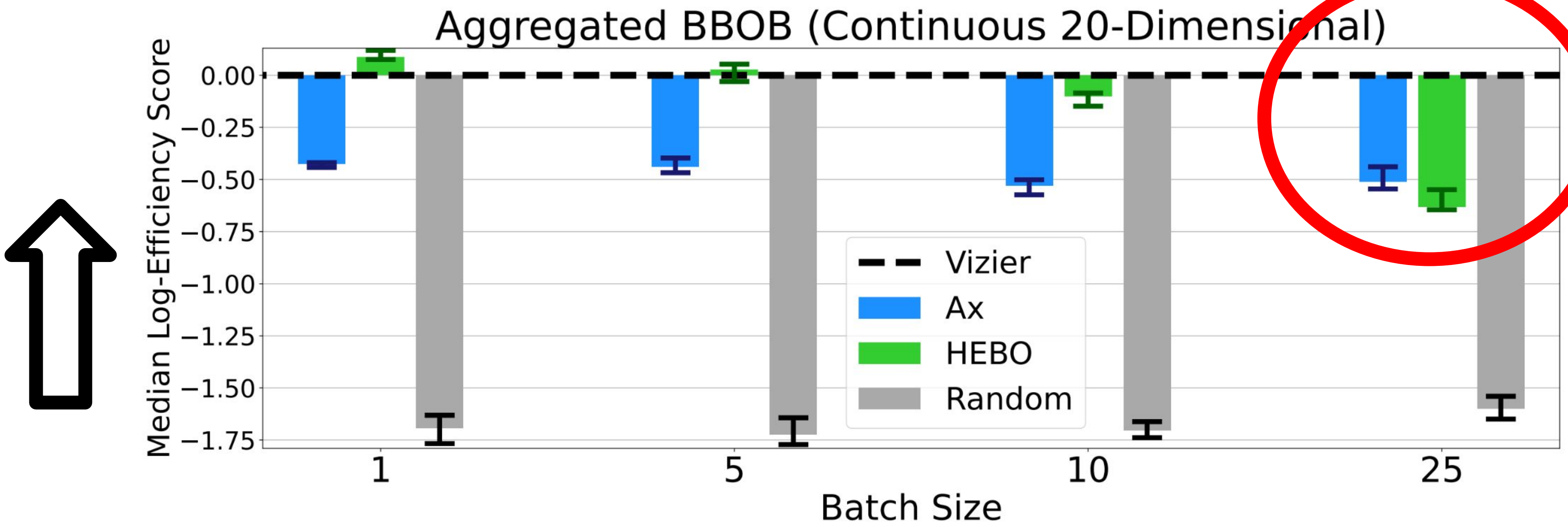
Batched Case



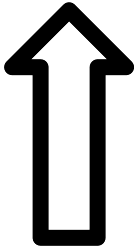


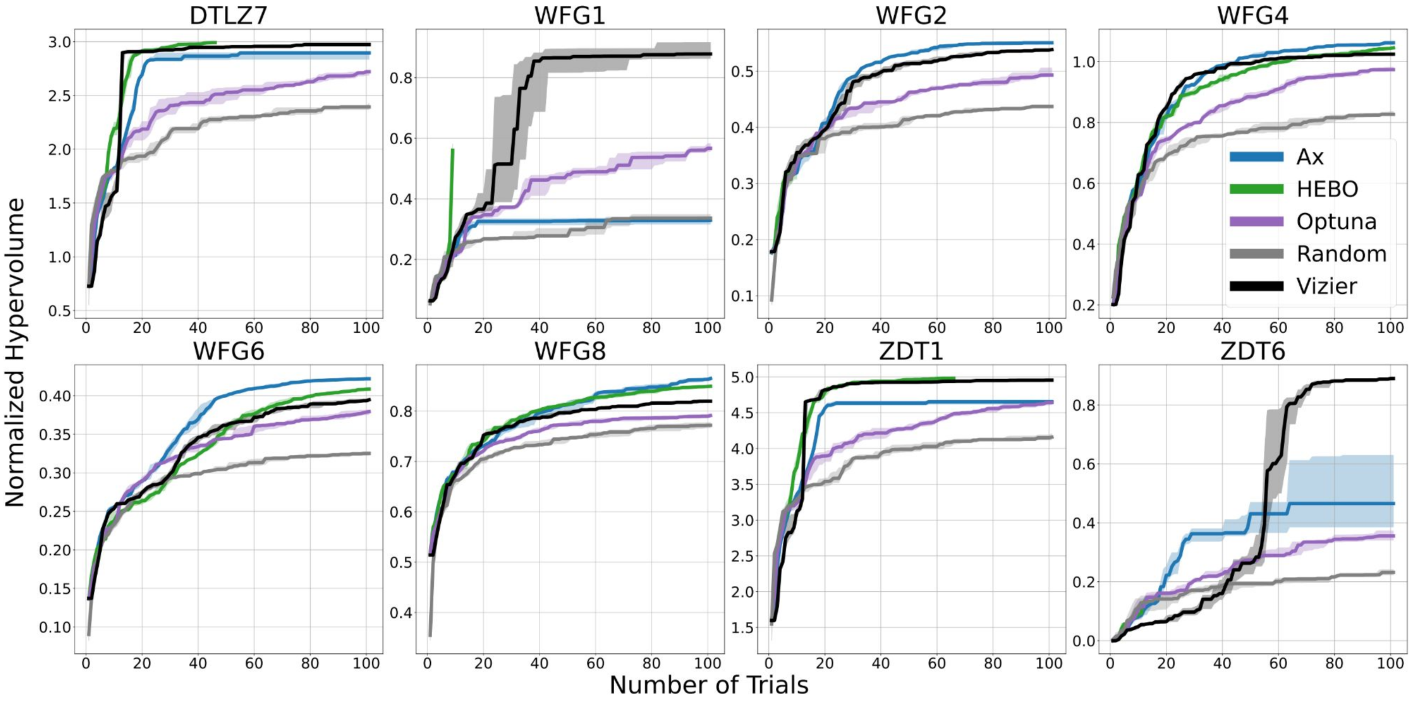
Batched Case

High batch settings still robust e.g. against HEBO



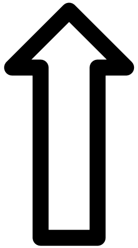
Multi-Objective (Individual Plots)

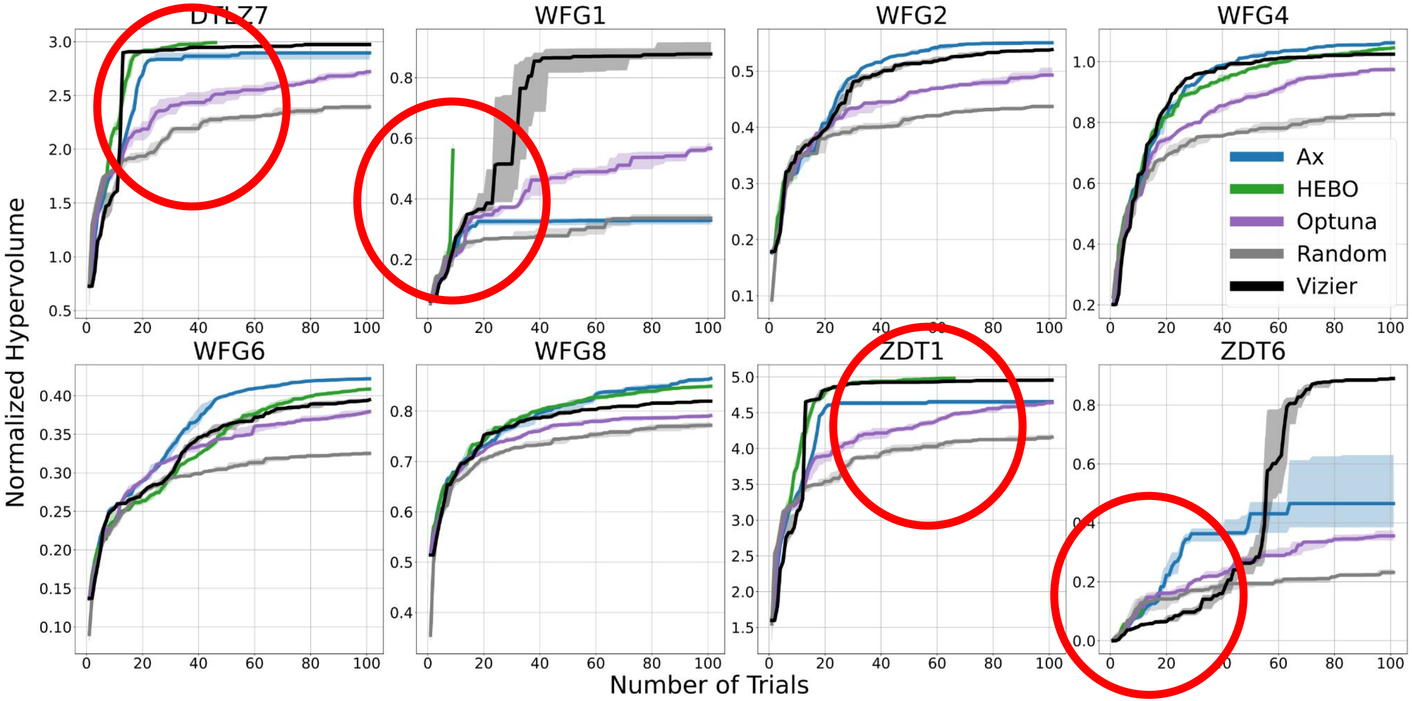




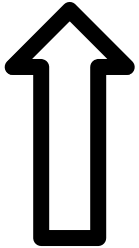
Multi-Objective (Individual Plots)

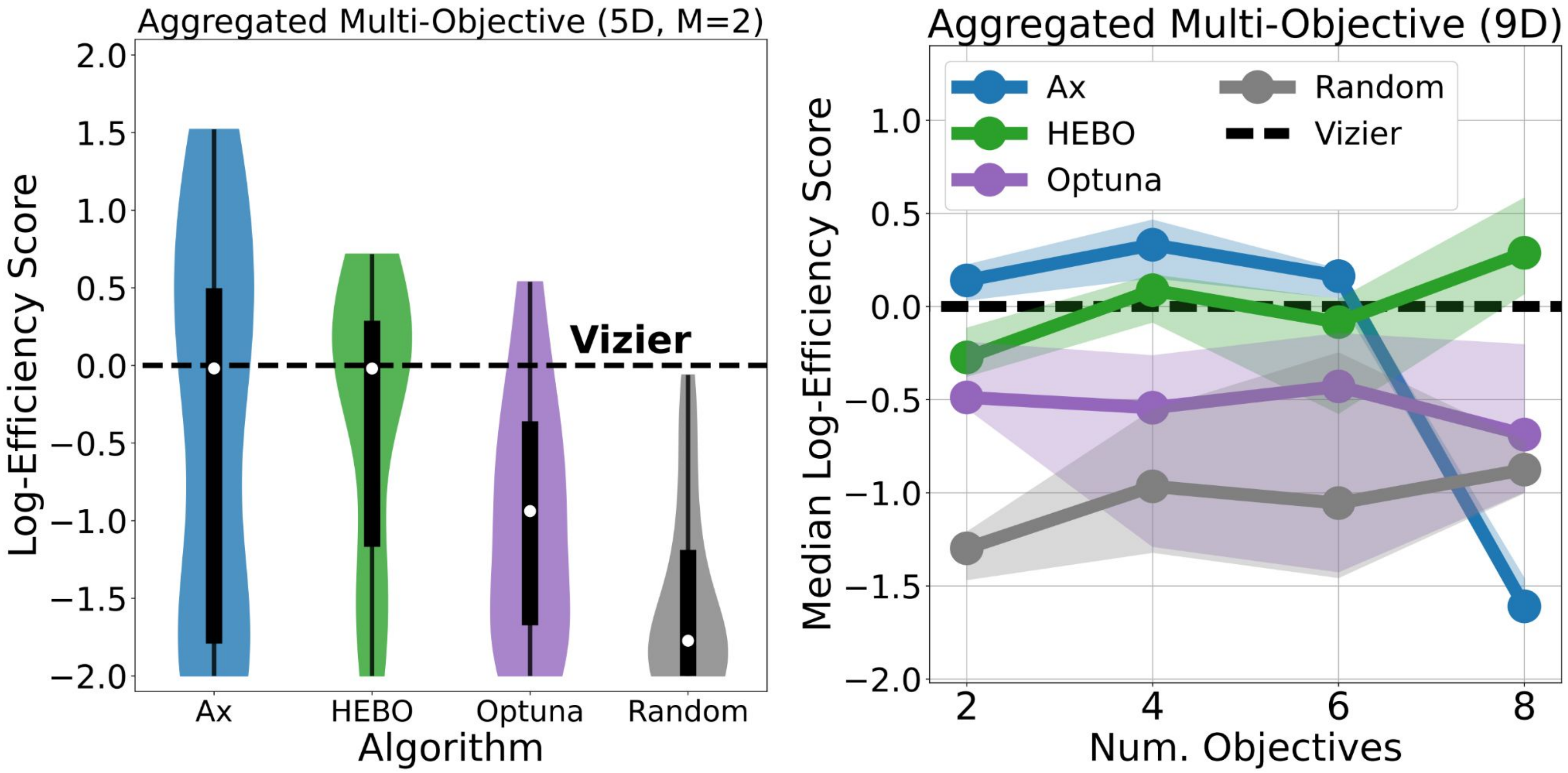
Instability with e.g. HEBO





Multi-Objective (Aggregate)

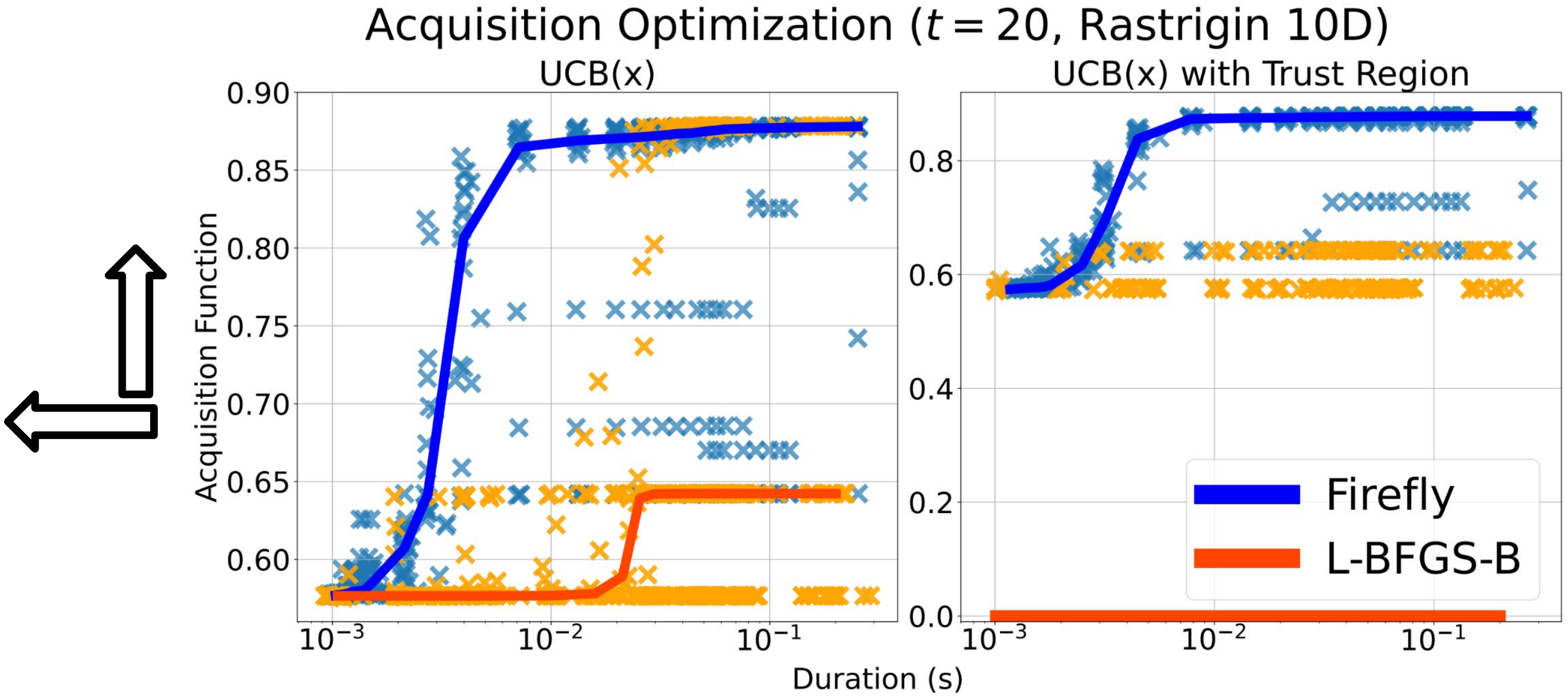




Questions?

Ablation: Acquisition Optimization

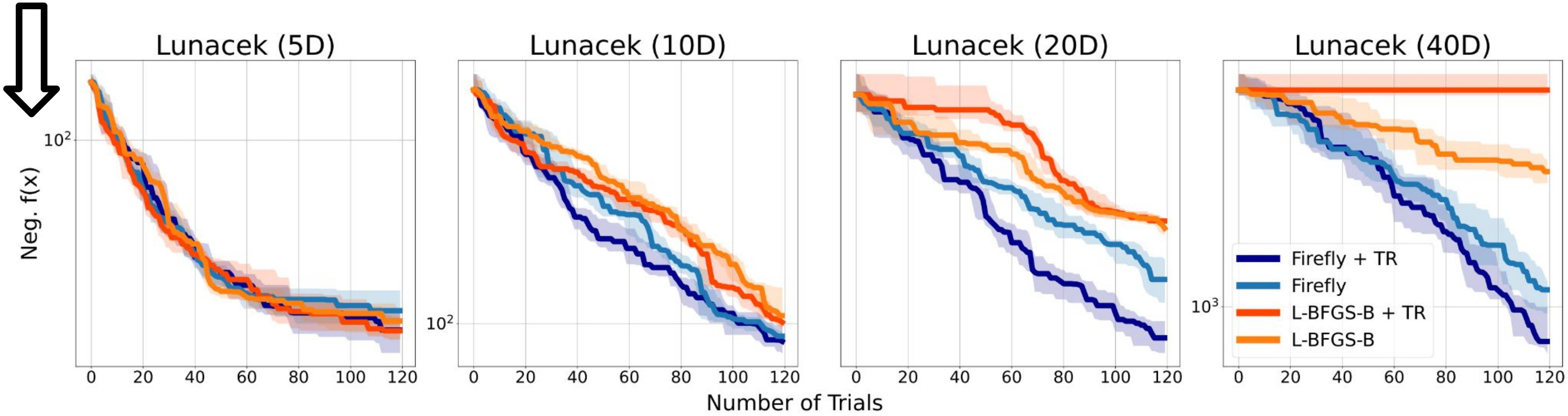
Firefly > L-BFGS-B



DeepMind

Ablation: AF-Optimization vs E2E

- Firefly > L-BFGS-B
- Trust Region (TR) > Without Trust Region

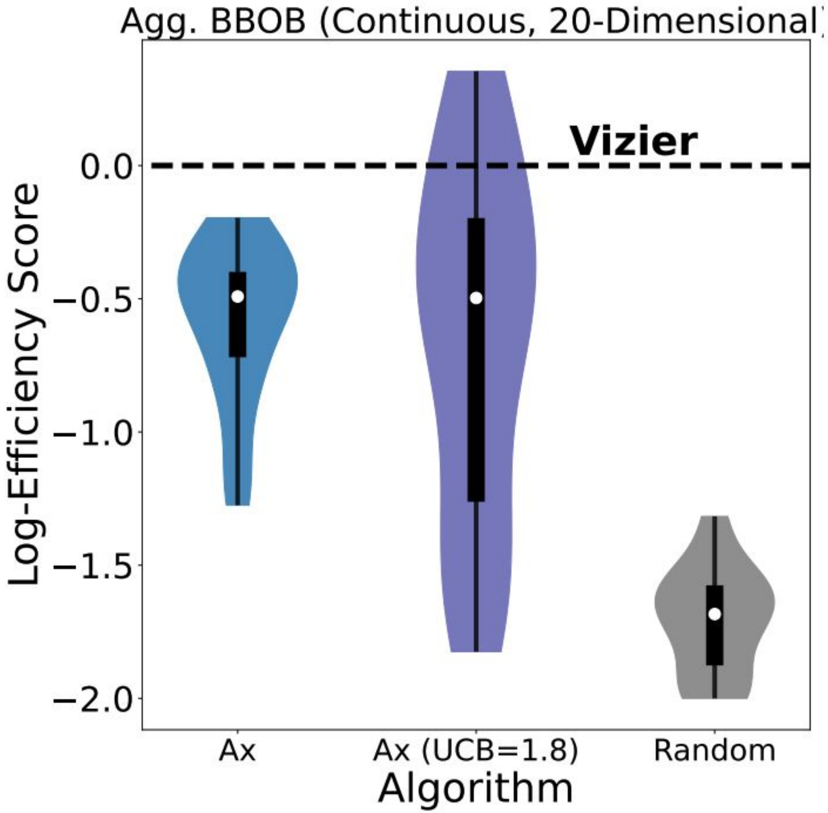


Ablation: What if Ax used UCB = 1.8?

Equalize acquisition functions.

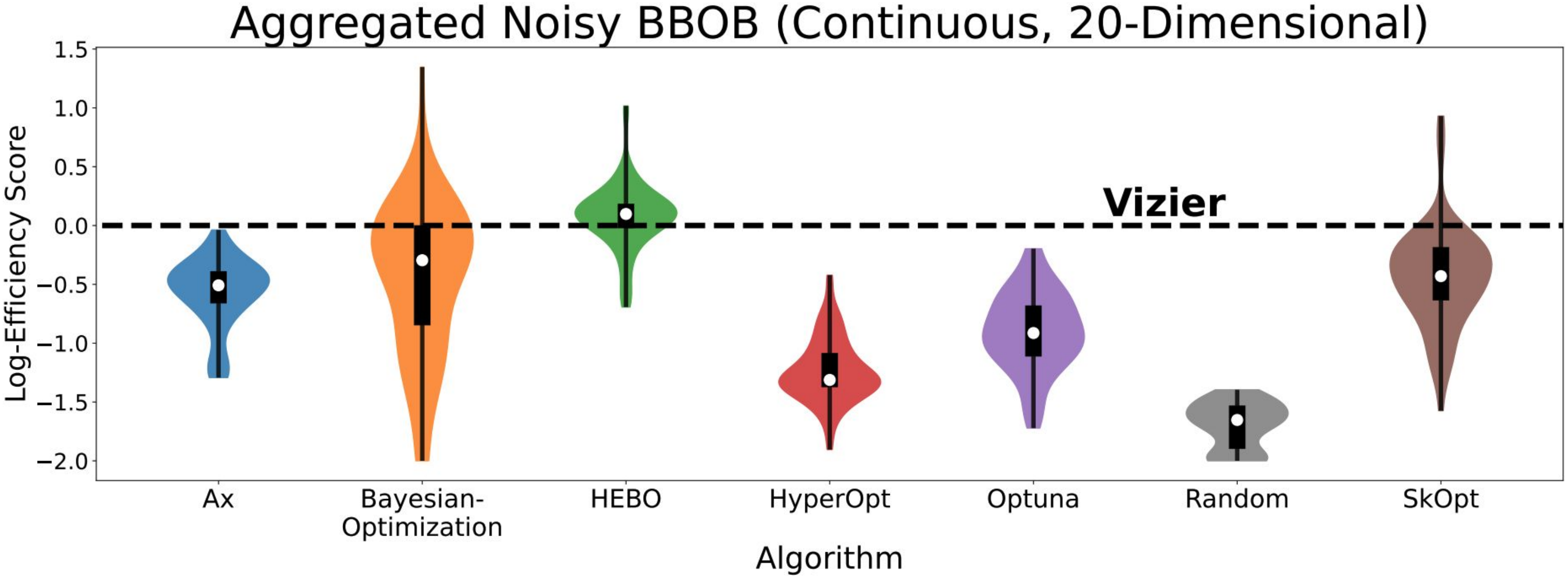
Ax's median still the same

Suggests AF optimizer is very important!



Ablation: Noisy Objectives

Ranking same as before.

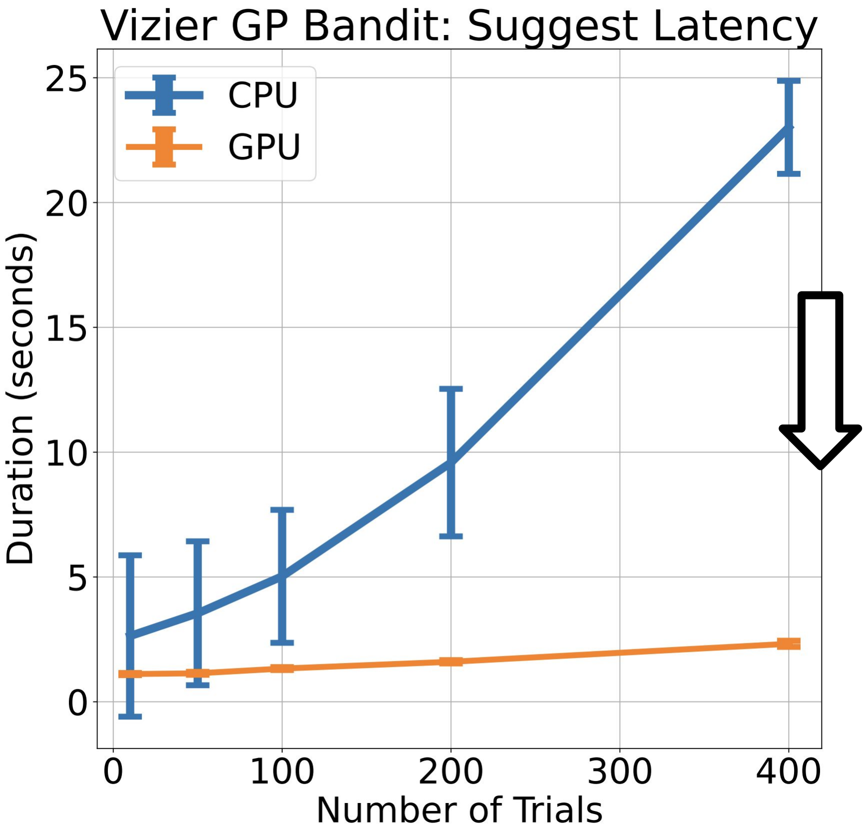


Google DeepMind

Ablation: Latency + GPU Acceleration

All components in JAX:

- Gaussian Process
- **Zero-Order Acq. Optimizer**
    - Others don't GPU-accelerate



Links

Code: https://github.com/google/vizier

Documentation: https://oss-vizier.readthedocs.io/en/latest/index.html

AI Blog:

https://ai.googleblog.com/2023/02/open-source-vizier-towards-reliable-and.html

Algorithm Paper: https://arxiv.org/abs/2408.11527

Systems Paper: https://arxiv.org/abs/2207.13676

Thanks!

Google DeepMind