LG전자 고급 데이터 사이언티스트 양성 과정 7기

Automated Machine Learning (AutoML)

서울대학교 컴퓨터공학부 이 영 기

Human-Centered Computer Systems Lab HCSLAB

SEOUL NATIONAL UNIVERSITY

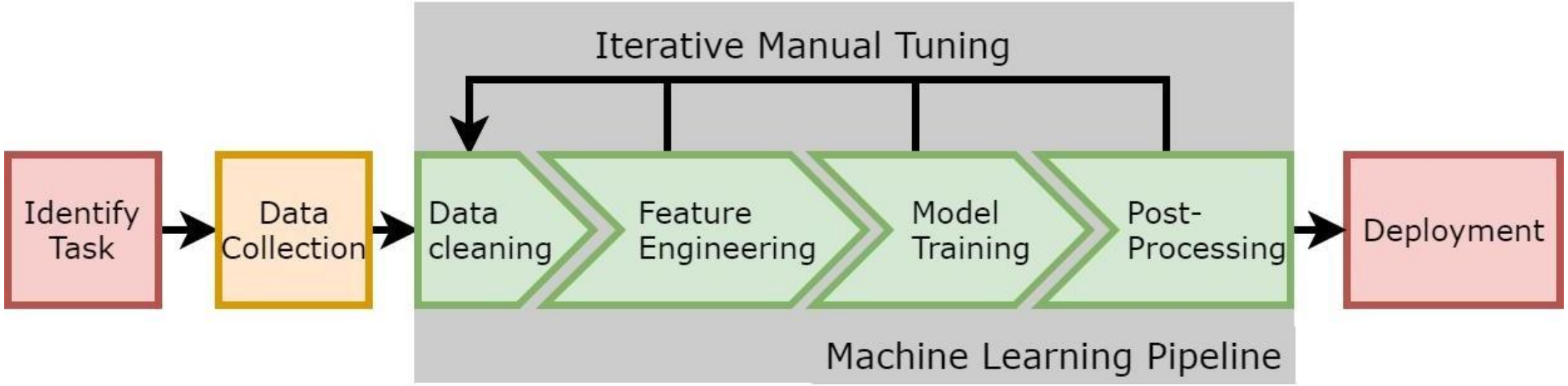
Overview

- Objective
  - Identifying challenges in traditional ML pipeline
  - Learn various Hyperparameter Optimization (HPO) techniques

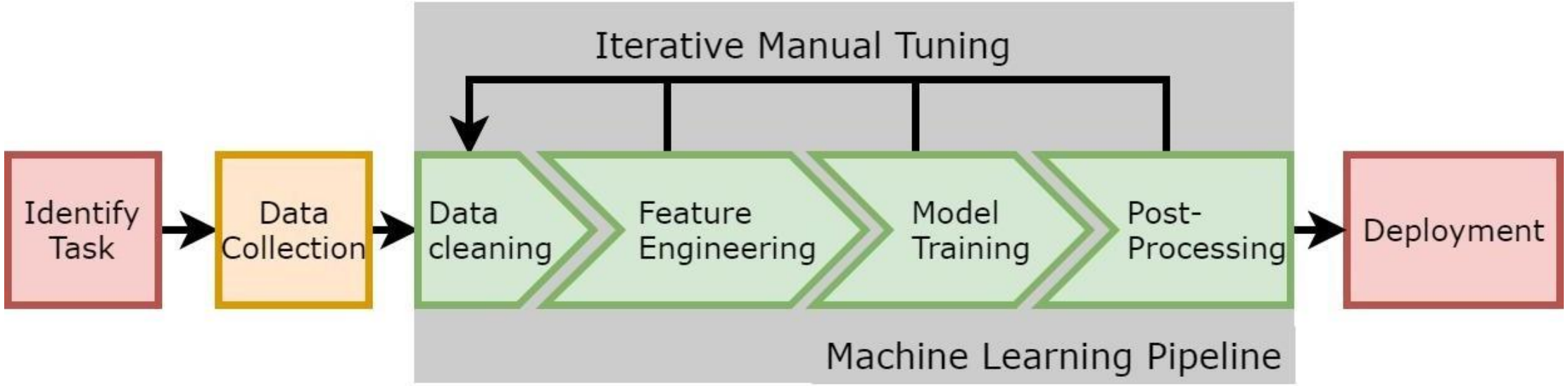
- Content
  - Introduction to AutoML
  - Hyperparameter Optimization
    - Blackbox Optimization(Grid search, Random search, Bayesian optimization, Tree-structured Parzen Estimator)
    - Multi-fidelity Optimization (Successive halving, Hyperband, Bayesian optimization + Hyperband)

Traditional Machine Learning Pipeline

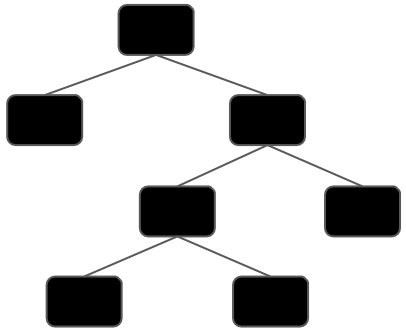
- Building an ML model is an iterative, complex, and time-consuming process
  - Clean & preprocess the data
  - Select / engineer better features
  - Select an algorithm
  - Set the model parameters
  - …



Challenges in Designing ML Pipelines



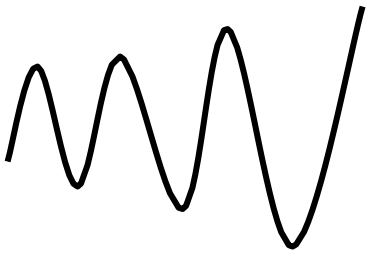
Complex Search Space



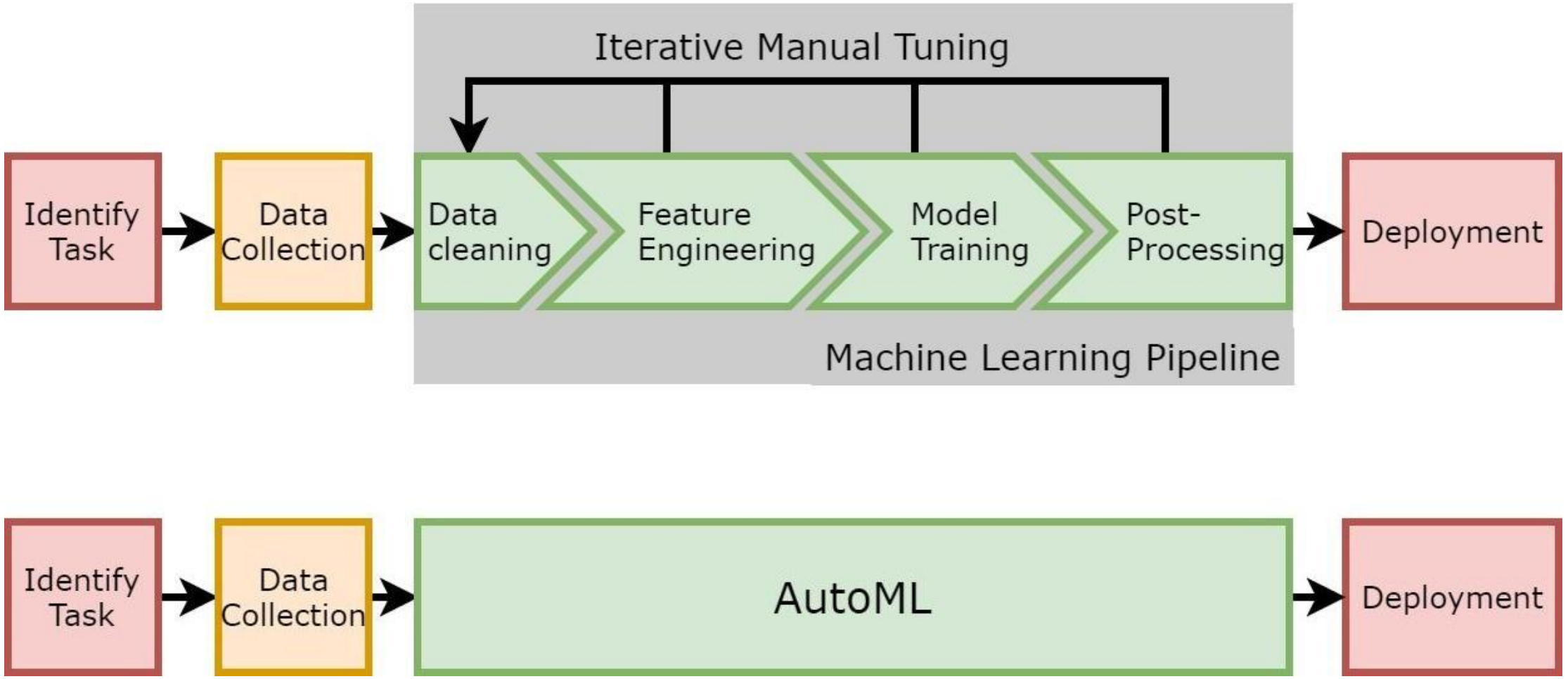
Black-Box Problem

Expensive Evaluations

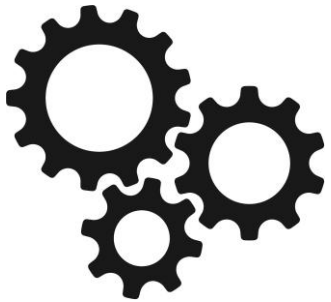
Noise on observations



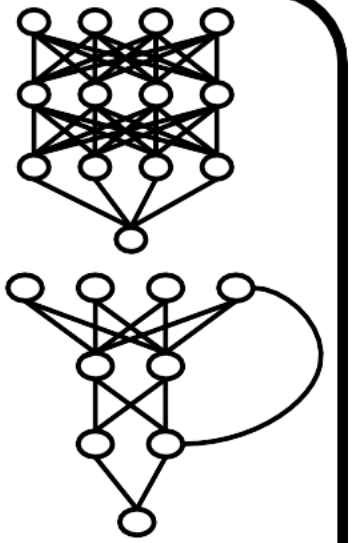
From Manual ML to Automated ML



Design Decisions taken care by AutoML



Hyper-parameters



Architecture Design

preprocessor	#λ
extreml. rand. trees prepr.	5
fast ICA	4
feature agglomeration	4
kernel PCA	5
rand. kitchen sinks	2
linear SVM prepr.	3
no preprocessing	-
nystroem sampler	5
PCA	2
polynomial	3
random trees embed.	4
select percentile	2
select rates	3
one-hot encoding	2
imputation	1
balancing	1
rescaling	1

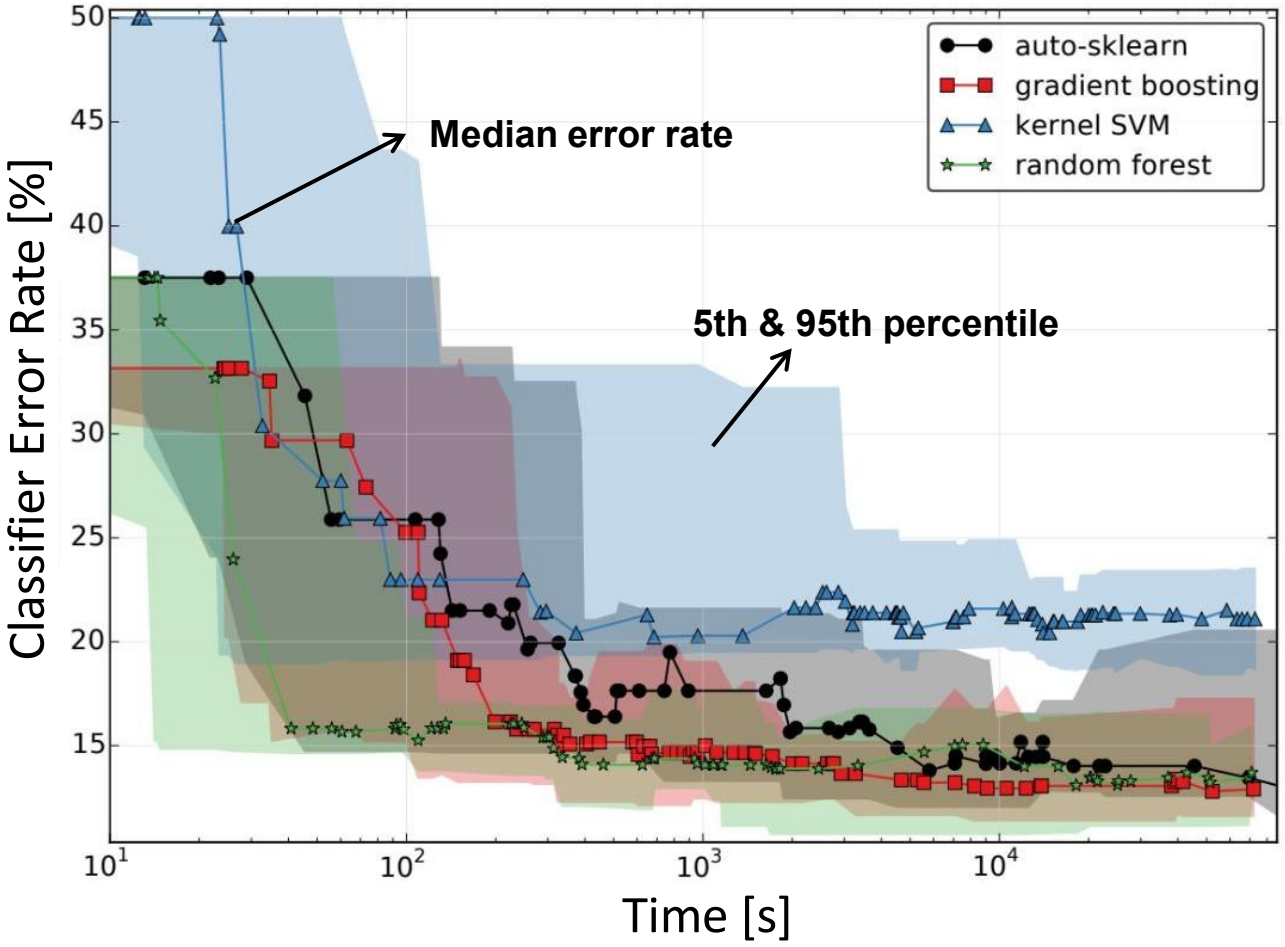
Pre-processing

classifier	#λ
AdaBoost (AB)	4
Bernoulli naïve Bayes	2
decision tree (DT)	4
extreml. rand. trees	5
Gaussian naïve Bayes	-
gradient boosting (GB)	6
kNN	3
LDA	4
linear SVM	4
kernel SVM	7
multinomial naïve Bayes	2
passive aggressive	3
QDA	2
random forest (RF)	5
Linear Class. (SGD)	10

Algorithms

Importance of Design Decisions in ML

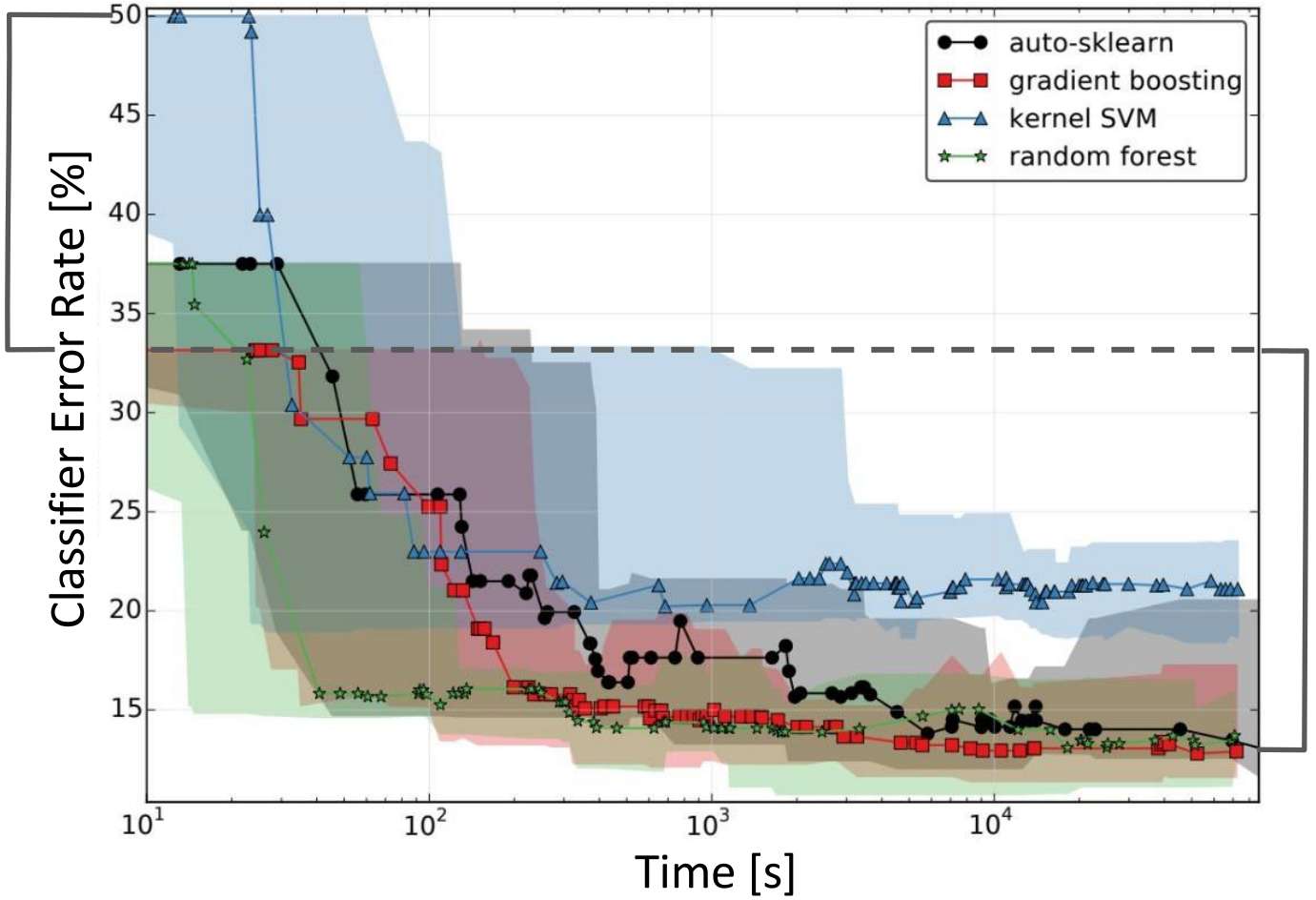
- Each decision is critical to overall performance
  - E.g. Classifiers performance compared to auto-sklearn over time in one OpenML dataset



Importance of Design Decisions in ML

- Each decision is critical to overall performance
  - E.g. Classifiers performance compared to auto-sklearn over time in one OpenML dataset

Choosing the gradient boosting algorithm instead of kernel SVM → 17% improv.



Optimizing hyperparameters in gradient boosting → 20% improv.

Hyperparameter Optimization (HPO)

Types of Hyperparameters

- Continuous
  - Regularization parameter C in SVM
  - Shrinkage parameter λ in boosting
- Integer
  - Number of trees in boosting and random forest
- Categorical
  - Algorithm (SVM, RF, NN, …)
  - SVM kernel (Polynomial, RBF, …)
  - Activation function (ReLU, Leaky ReLU, …)

Types of Hyperparameters

- Conditional: hyperparameters B are only active if other hyperparameters A are set a certain way
  - Example 1:
    - A = choice of optimizer (Adam or SGD)
    - B = Adam‘s second momentum hyperparameter (only active if A=Adam)
  - Example 2:
    - A = type of layer k (convolution, max pooling, fully connected, ...)
    - B = conv. kernel size of that layer (only active if A = convolution)
  - Example 3:
    - A = choice of classifier (RF or SVM)
    - B = SVM‘s kernel parameter (only active if A = SVM)

AutoML as Hyperparameter Optimization

Definition: Combined Algorithm Selection and Hyperparameter Optimization (CASH)

Let

- $\mathcal{A} = \{A^{(1)}, \ldots, A^{(n)}\}$ be a set of algorithms
- $\mathbf{\Lambda}^{(i)}$ denote the hyperparameter space of $A^{(i)}$, for $i = 1, \ldots, n$
- $\mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of $A^{(i)}$, using $\lambda \in \mathbf{\Lambda}^{(i)}$ trained on D_{train} and evaluated on D_{valid}.

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem is to find a combination of algorithm $A^* = A^{(i)}$ and hyperparameter configuration $\boldsymbol{\lambda}^* \in \mathbf{\Lambda}^{(i)}$ that minimizes this loss:

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A^{(i)} \in \mathcal{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}^{(i)}}{\arg\min} \mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

→ Simply a HPO problem with a top-level hyperparameter (choice of algorithm) that all other hyperparameters are conditional on.

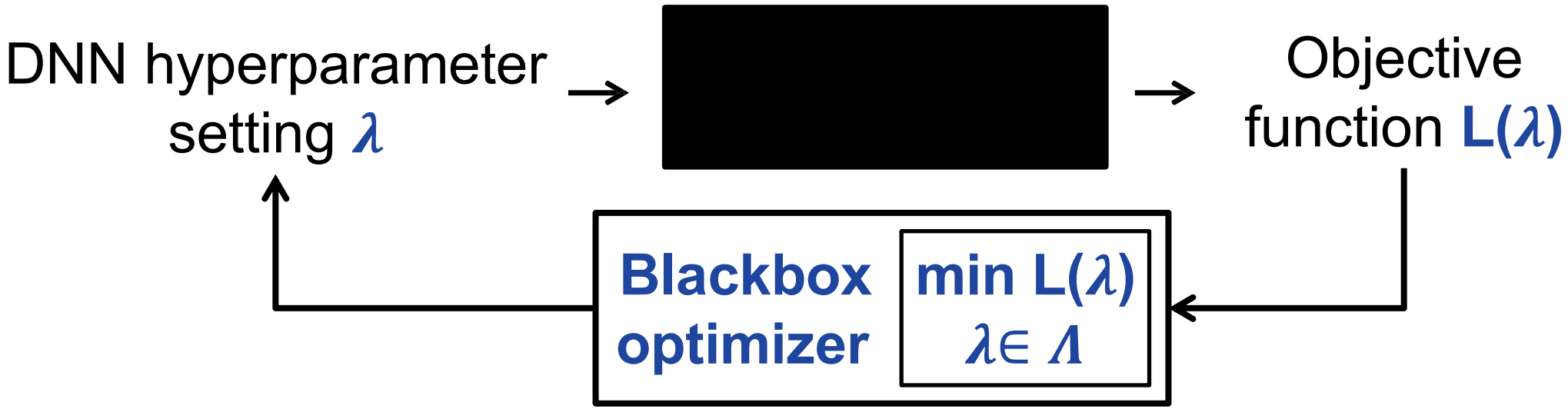
Hyperparameter Optimization

- Blackbox optimization
  - Grid search
  - Random search
  - Bayesian optimization
  - Tree-structured Parzen Estimator

- Multi-fidelity optimization
  - Successive Halving
  - Hyperband
  - Bayesian optimization + Hyperband

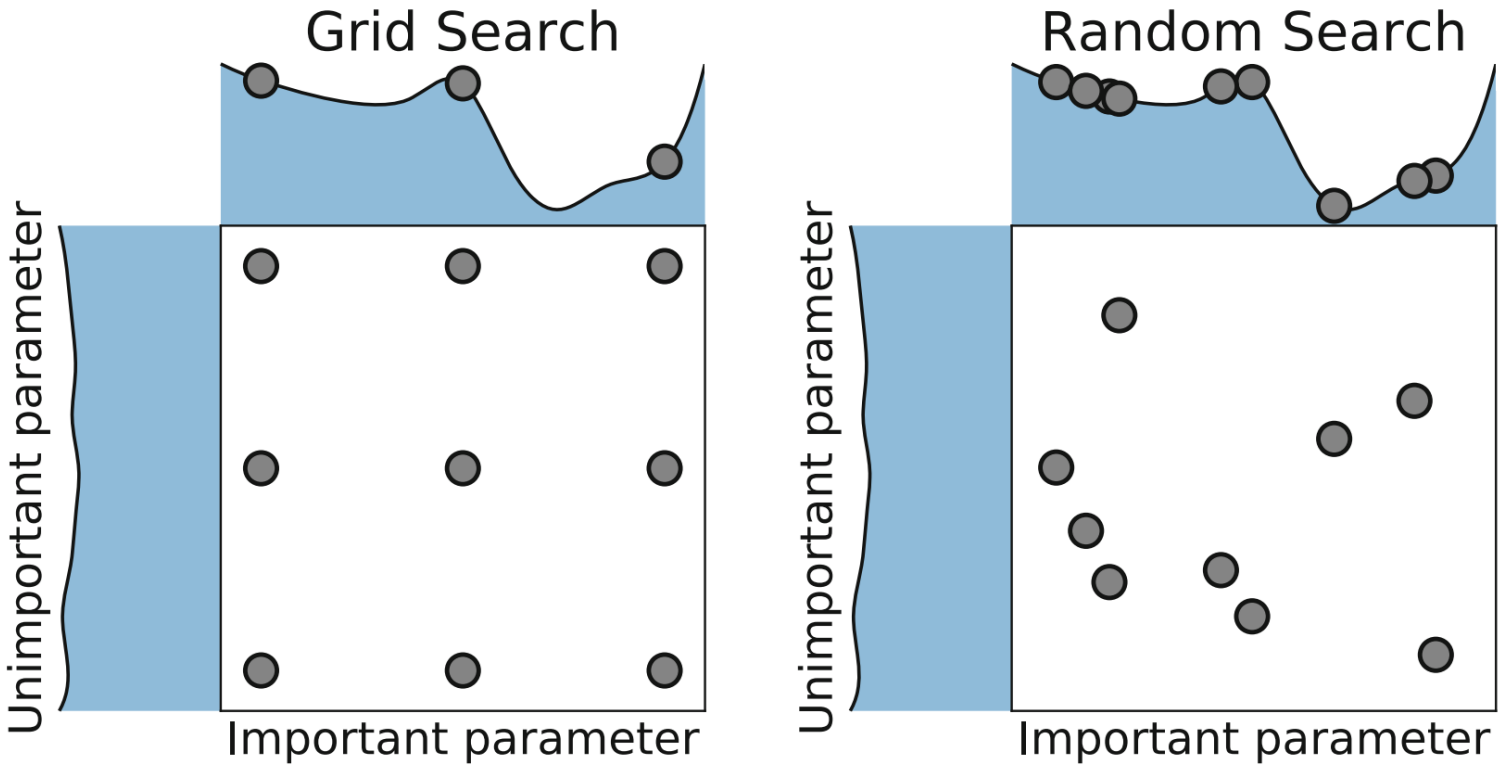
Blackbox Hyperparameter Optimization

- The blackbox function is expensive to evaluate
  → Can't sample too often!



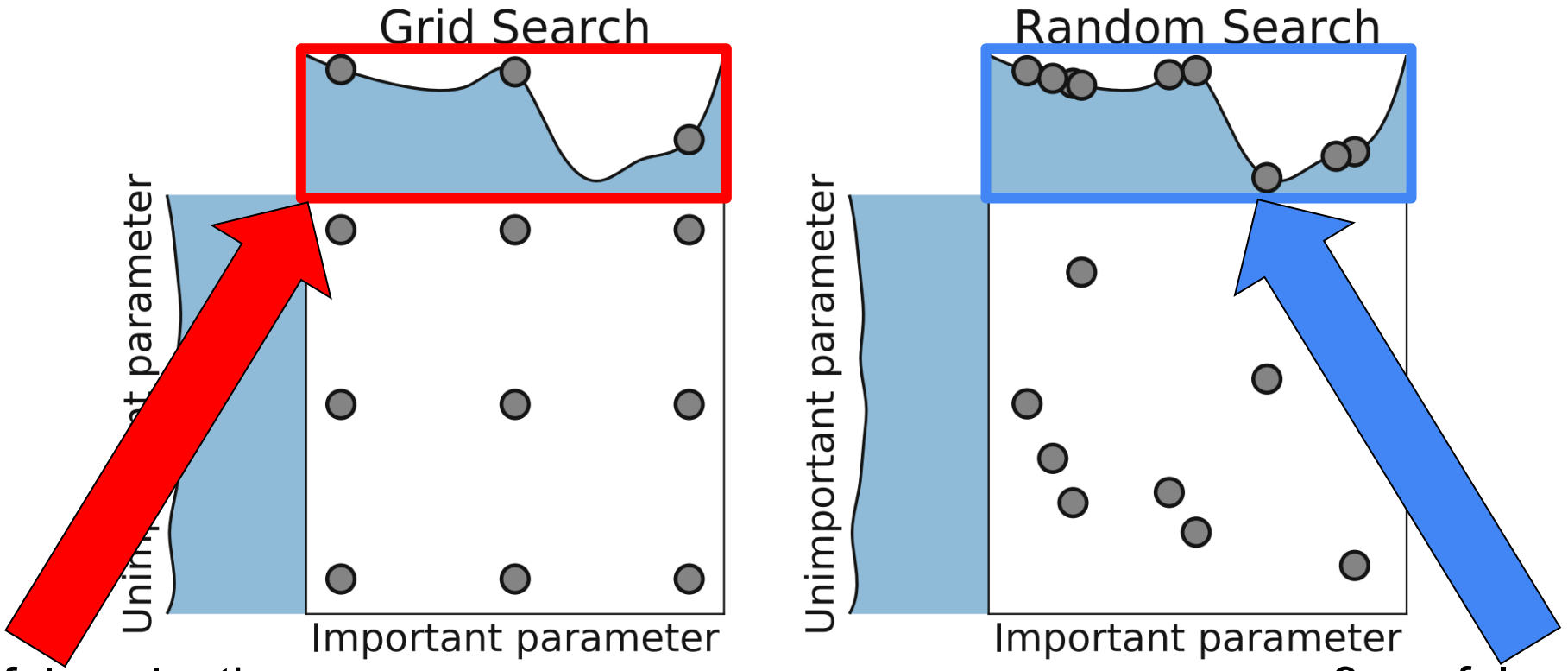
Blackbox Optimization: Grid and Random Search

- Grid search: required number of samples grows exponentially
- Random search: handles unimportant dimensions better



Blackbox Optimization: Grid and Random Search

- Grid search: required number of samples grows exponentially
- Random search: handles unimportant dimensions better



Only 3 useful evaluations

9 useful evaluations

Bayesian Optimization

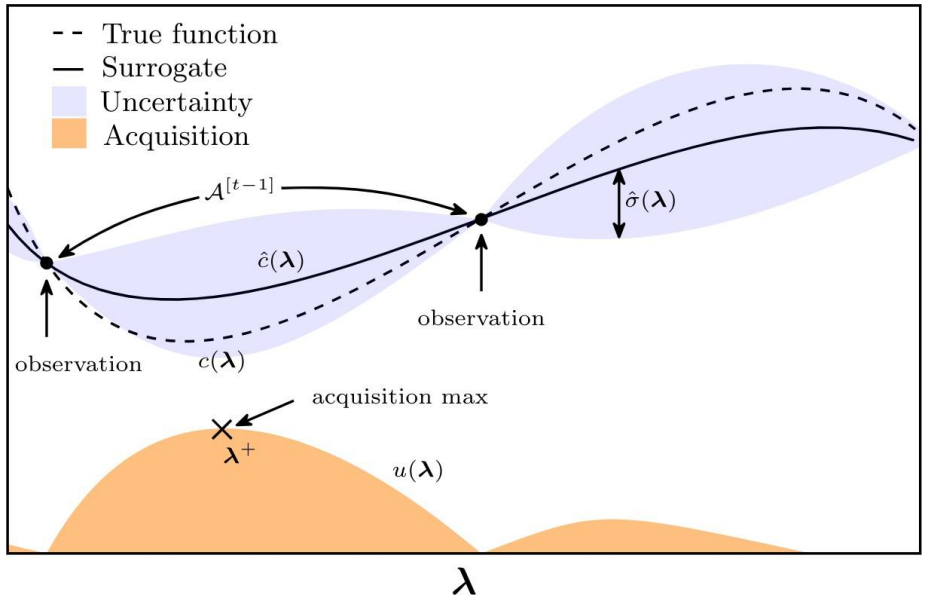
- Limitation of previous techniques: completely uninformed and does not learn about the search space
- If we have already evaluated a set of candidates, can’t we do better than random selection?

=> Predict promising regions based on history of observations

Bayesian Optimization

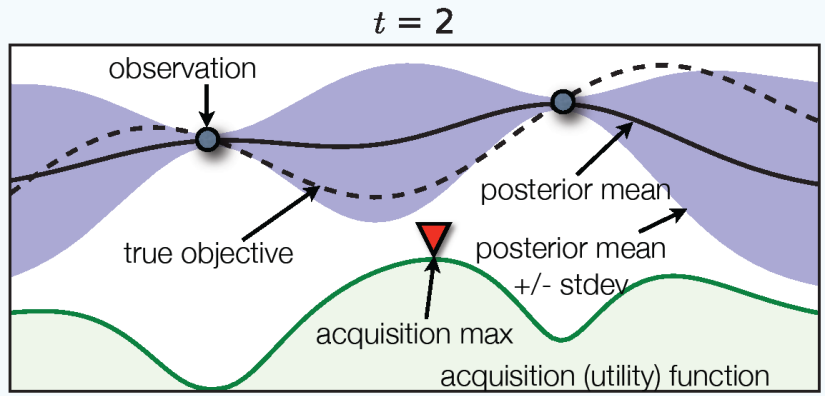
- Update the surrogate model sequentially with new observations
- Use the acquisition function to select new promising configurations to evaluate
- Surrogate Model
  - Probabilistic modeling of $\mathrm{L}(\lambda) \sim (\hat{\mathrm{L}}(\lambda), \hat{\sigma}(\lambda))$ with posterior mean $\hat{\mathrm{L}}(\lambda)$ and uncertainty $\hat{\sigma}(\lambda)$
    E.g. Gaussian Processes

- Acquisition Function
  - Balance exploration (high $\hat{\sigma}$) vs. exploitation (low $\hat{\mathrm{L}}$)
  - E.g. Expected improvement (EI):
    $u(\boldsymbol{\lambda}) = \mathrm{E}[\max\{\mathrm{L}_{\min} - \mathrm{L}(\boldsymbol{\lambda}), 0\}]$, where $L_{\min}$ is lowest value from observations



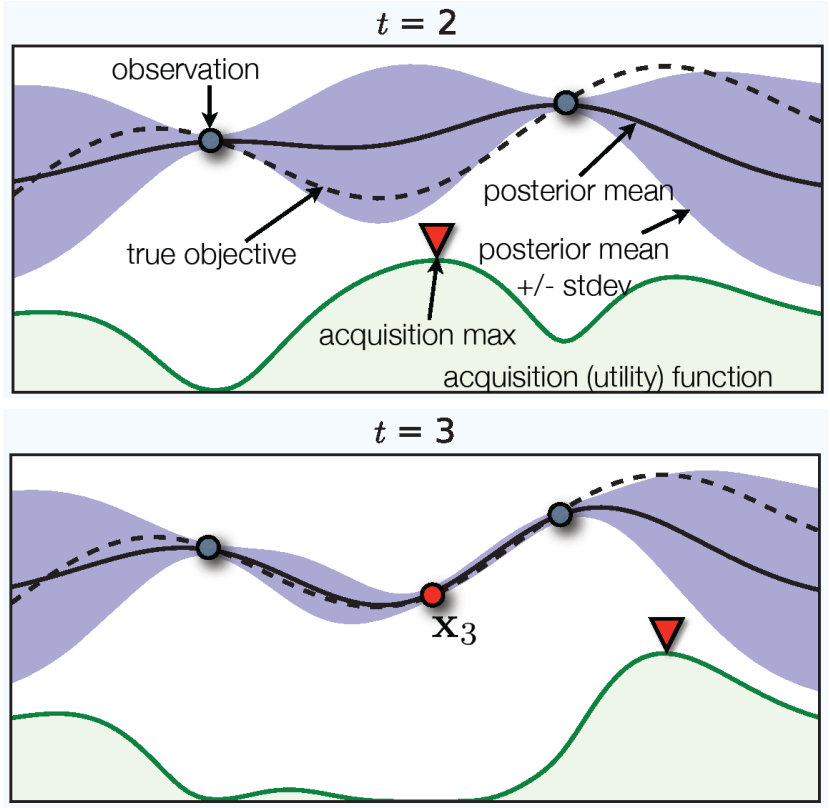
Bayesian Optimization

1. Start with a few random hyperparameter configurations
2. Fit a surrogate model to observations
3. Compute the acquisition function and select new configurations accordingly
4. Evaluate new points and get observations
5. Repeat 2-4



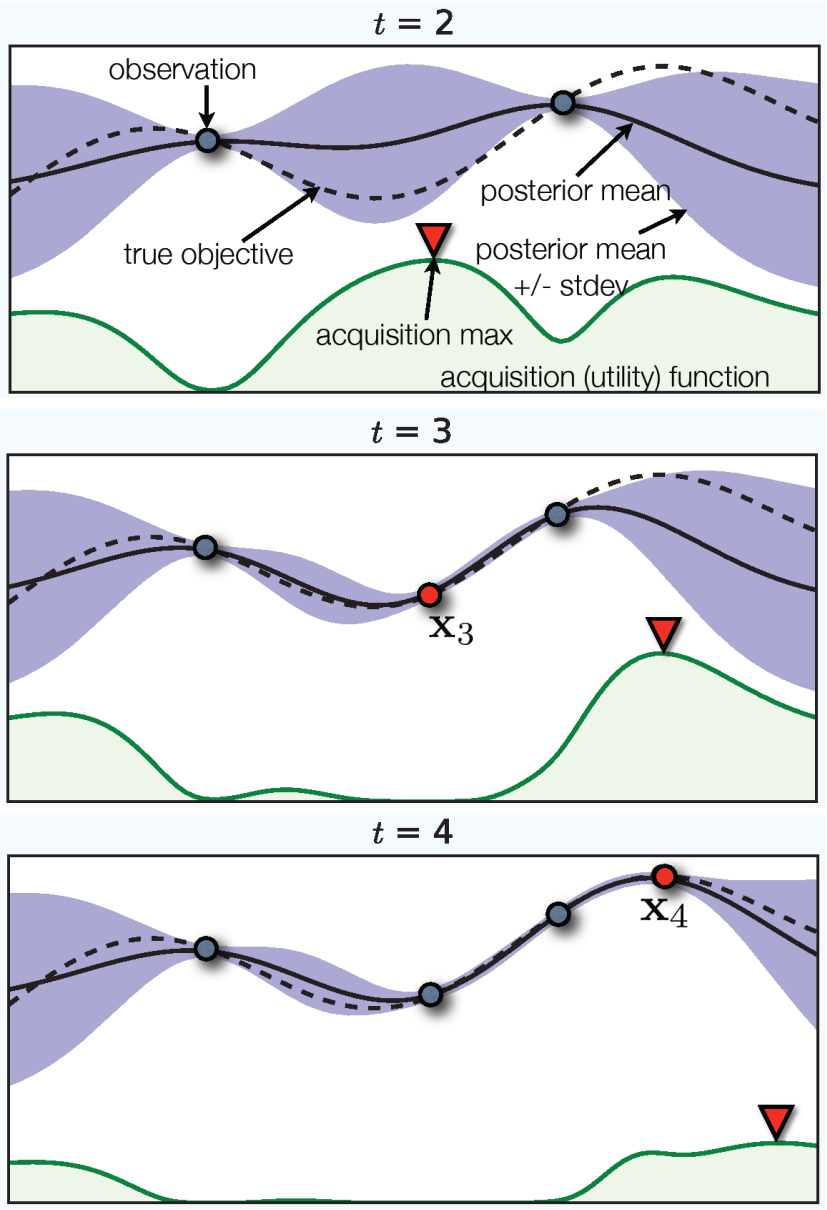
Bayesian Optimization

1. Start with a few random hyperparameter configurations
2. Fit a surrogate model to observations
3. Compute the acquisition function and select new configurations accordingly
4. Evaluate new points and get observations
5. Repeat 2-4

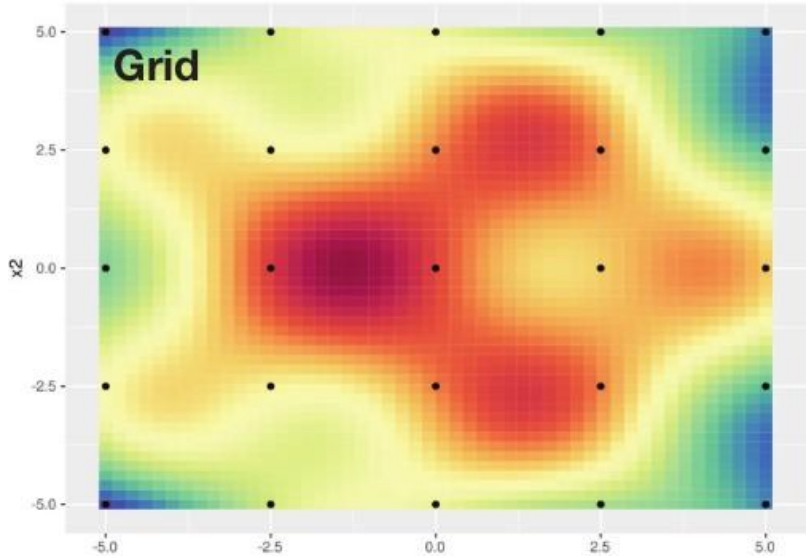


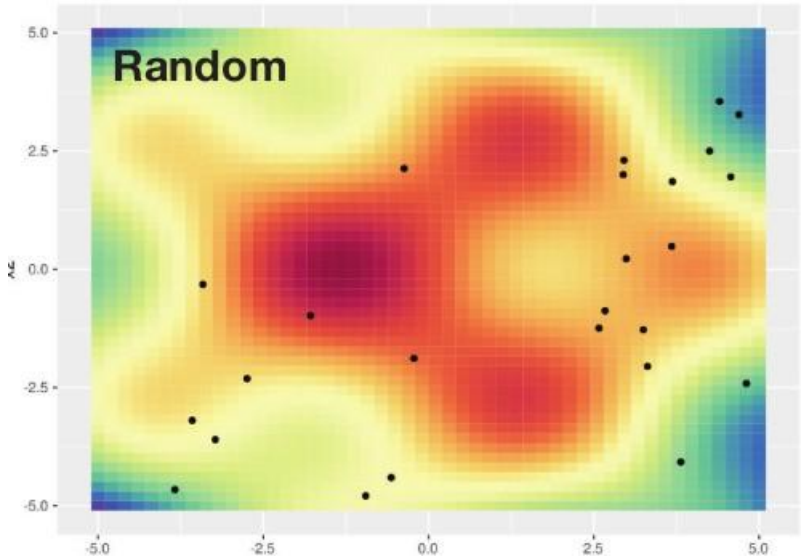
Bayesian Optimization

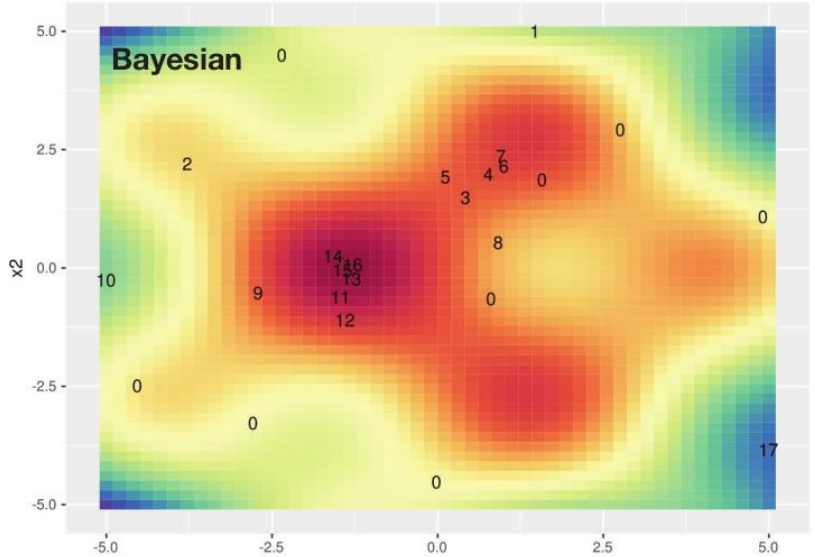
1. Start with a few random hyperparameter configurations
2. Fit a surrogate model to observations
3. Compute the acquisition function and select new configurations accordingly
4. Evaluate new points and get observations
5. Repeat 2-4



Bayesian Optimization

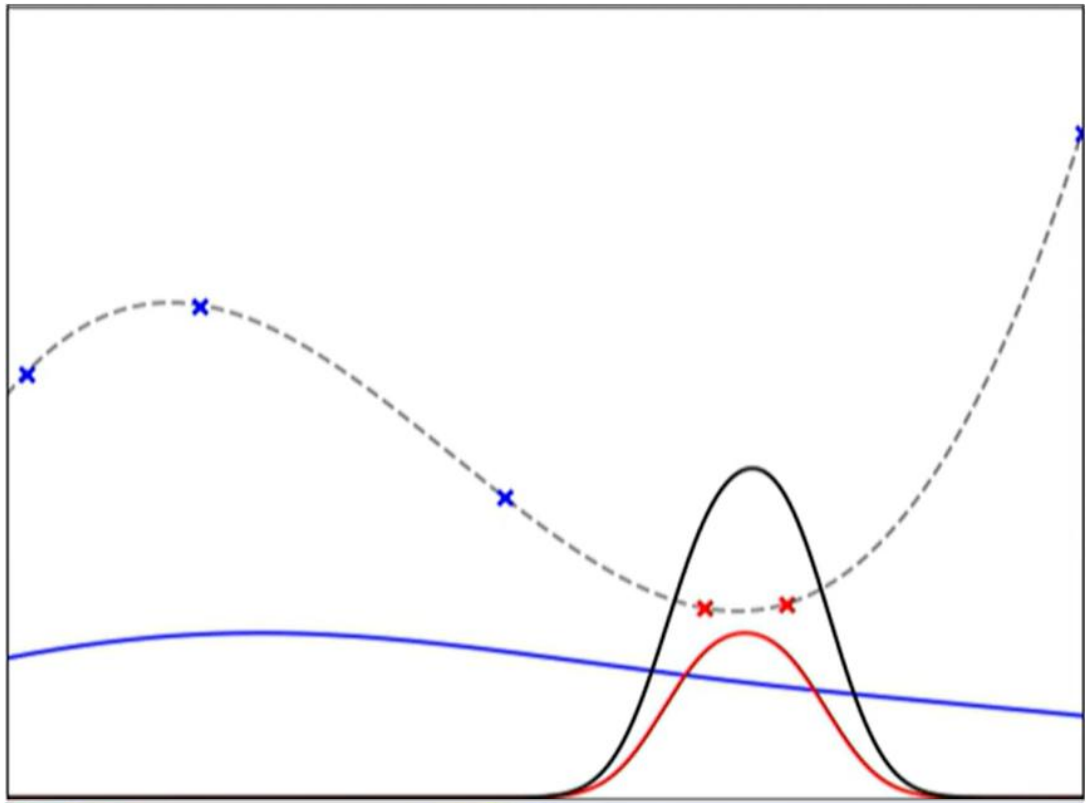






Tree-structured Parzen Estimator (TPE)

- Problems for standard Gaussian Process approach:
  - Not scalable to high dimensional hyperparameter spaces
  - Unfit for discrete spaces since GP assumes smoothness (e.g. Conditional hyperparameters)
- Maintain two surrogate models
  p(λ is good) and p(λ is bad),
  rather than p(L | λ)
  - P(λ is good) = P(λ | L ≤ L*)
  - P(λ is bad) = P(λ | L > L*)
    (L*: split threshold)
- Acquisition function $\propto \frac{P(\lambda \text{ is good})}{P(\lambda \text{ is bad})}$
  - Equivalent to expected improvement



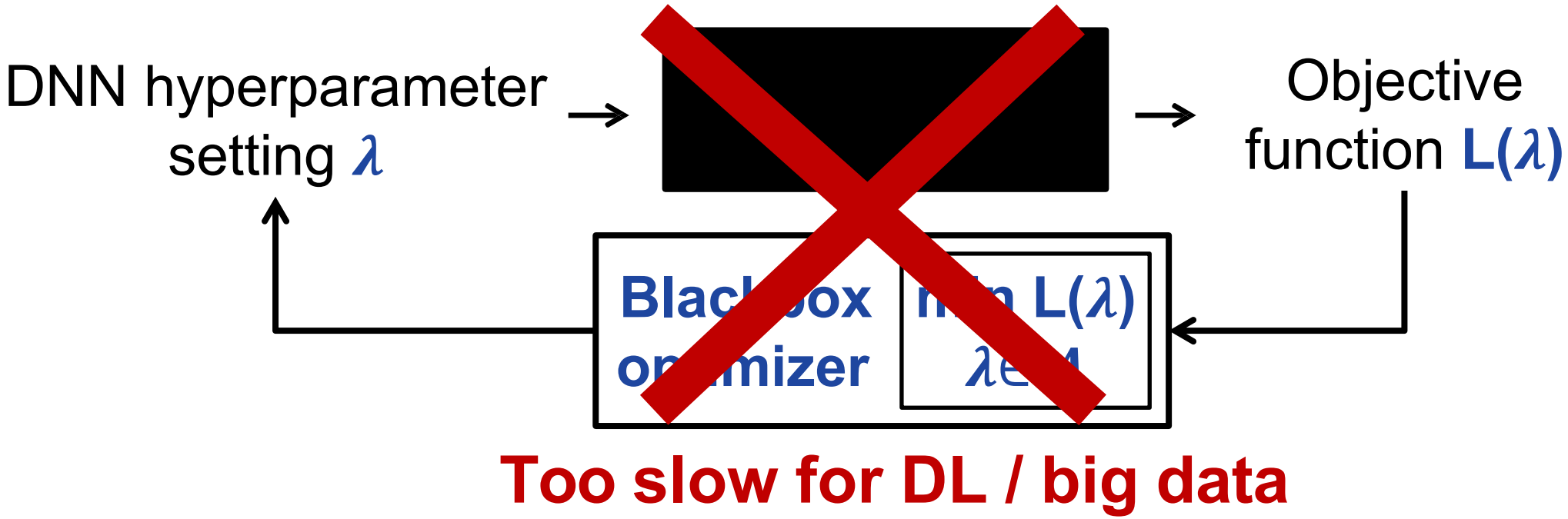
Hyperparameter Optimization

- Blackbox optimization
  - Grid search
  - Random search
  - Bayesian optimization
  - Tree-structured Parzen Estimator

- Multi-fidelity optimization
  - Successive Halving
  - Hyperband
  - Bayesian optimization + Hyperband

Multi-fidelity Optimization

- Massive dataset sizes and complex models make blackbox performance evaluation expensive

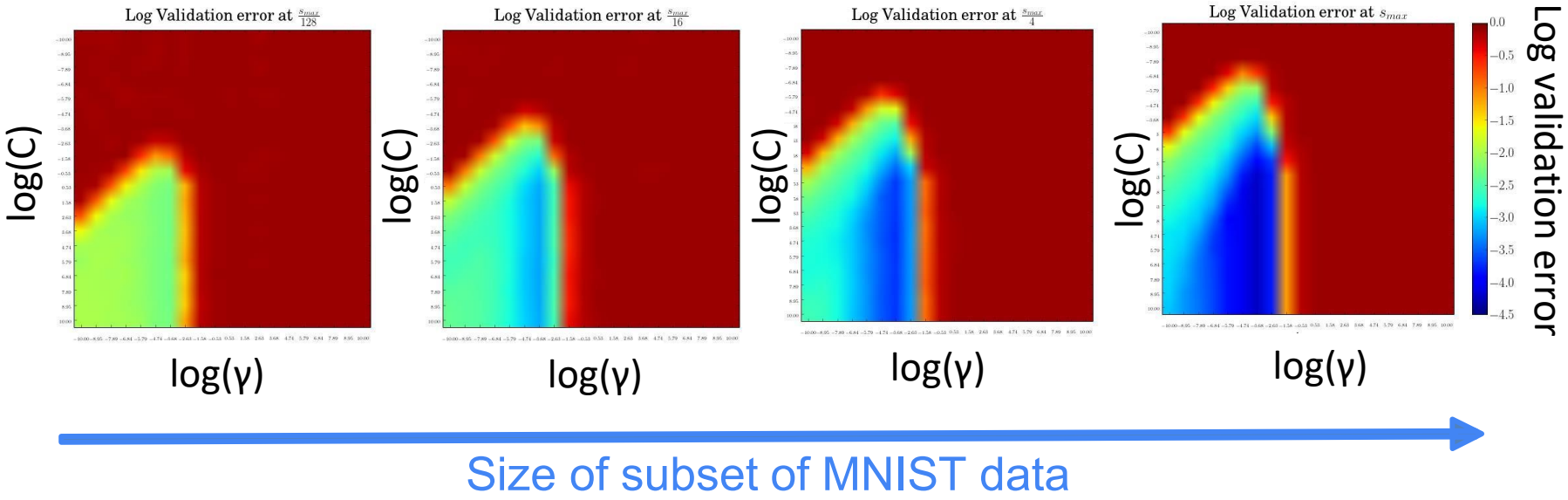


Multi-fidelity Optimization

- Massive dataset sizes and complex models make blackbox performance evaluation expensive
- Make use of cheap low-fidelity evaluations with a small budget, e.g.
  - Subsets of the data
  - Fewer epochs of iterative training algorithms
- These approximations introduce a tradeoff between optimization performance and runtime

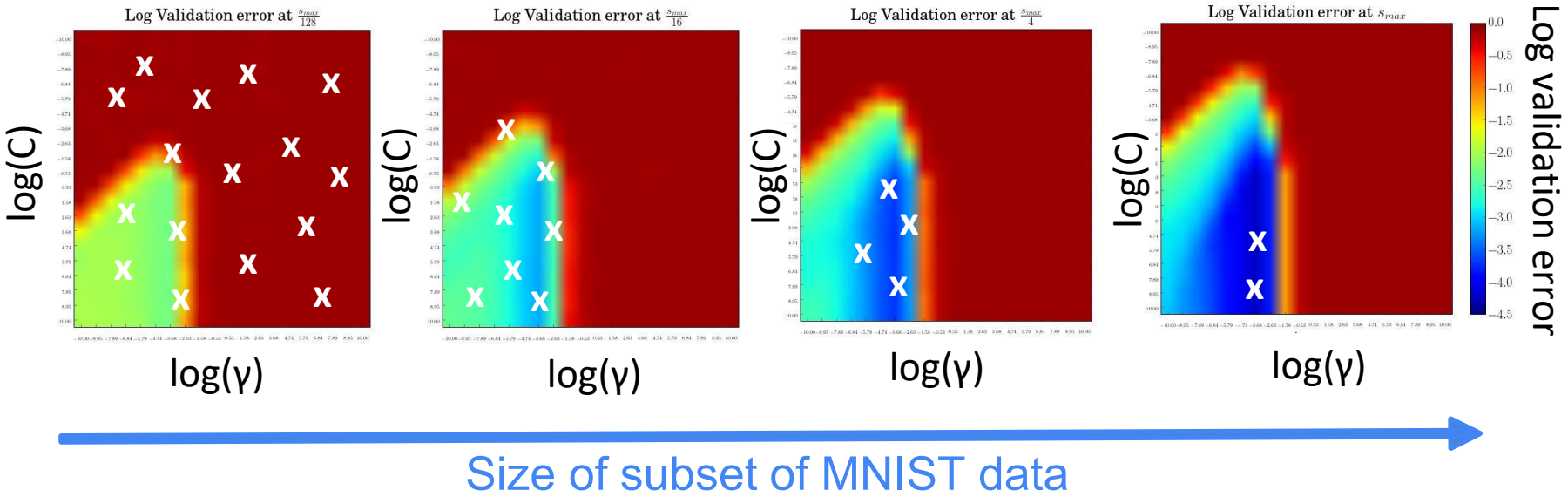
Successive Halving

- Many cheap evaluations on small subsets & few expensive evaluations on the full data
- E.g. cheap low-fidelity evaluation of SVM, with subsets of the MNIST data



Successive Halving

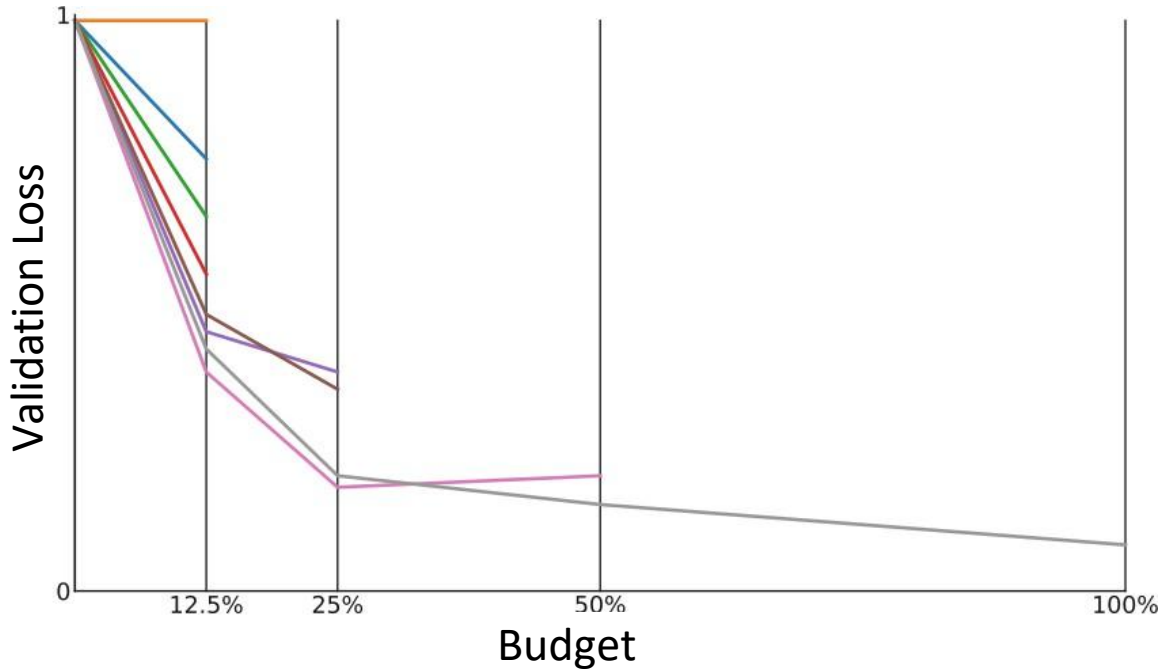
- Many cheap evaluations on small subsets & few expensive evaluations on the full data
- E.g. cheap low-fidelity evaluation of SVM, with subsets of the MNIST data



- Up to 1000x speedups

Successive Halving

- Successive halving for eight algorithms/configurations
- After evaluating all algorithms on 1/8 of the total budget, half of them are dropped and the budget given to the remaining algorithms is doubled

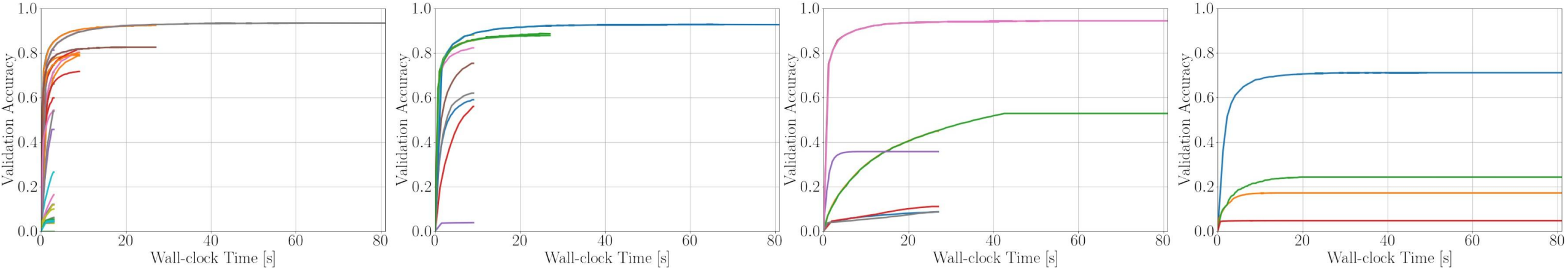


Successive Halving: Limitation

- Budget vs. number tradeoff in SHA
  - Given a total budget, the user has to decide whether:
    - Try many configurations and only assign a small budget to each
    - Try only a few and assign them a larger budget
- Assigning too small budget can result in prematurely terminating good configurations
- Assigning too large budget can result in running poor configurations too long

HyperBand

- Trying various budget-number combinations
  - Repeat SHA with different initial number of configurations n and start budgets m
  - Randomly select the initial configurations for each iteration
  - Return configuration with the smallest intermediate loss seen so far



High n & Low m

Low n & High m

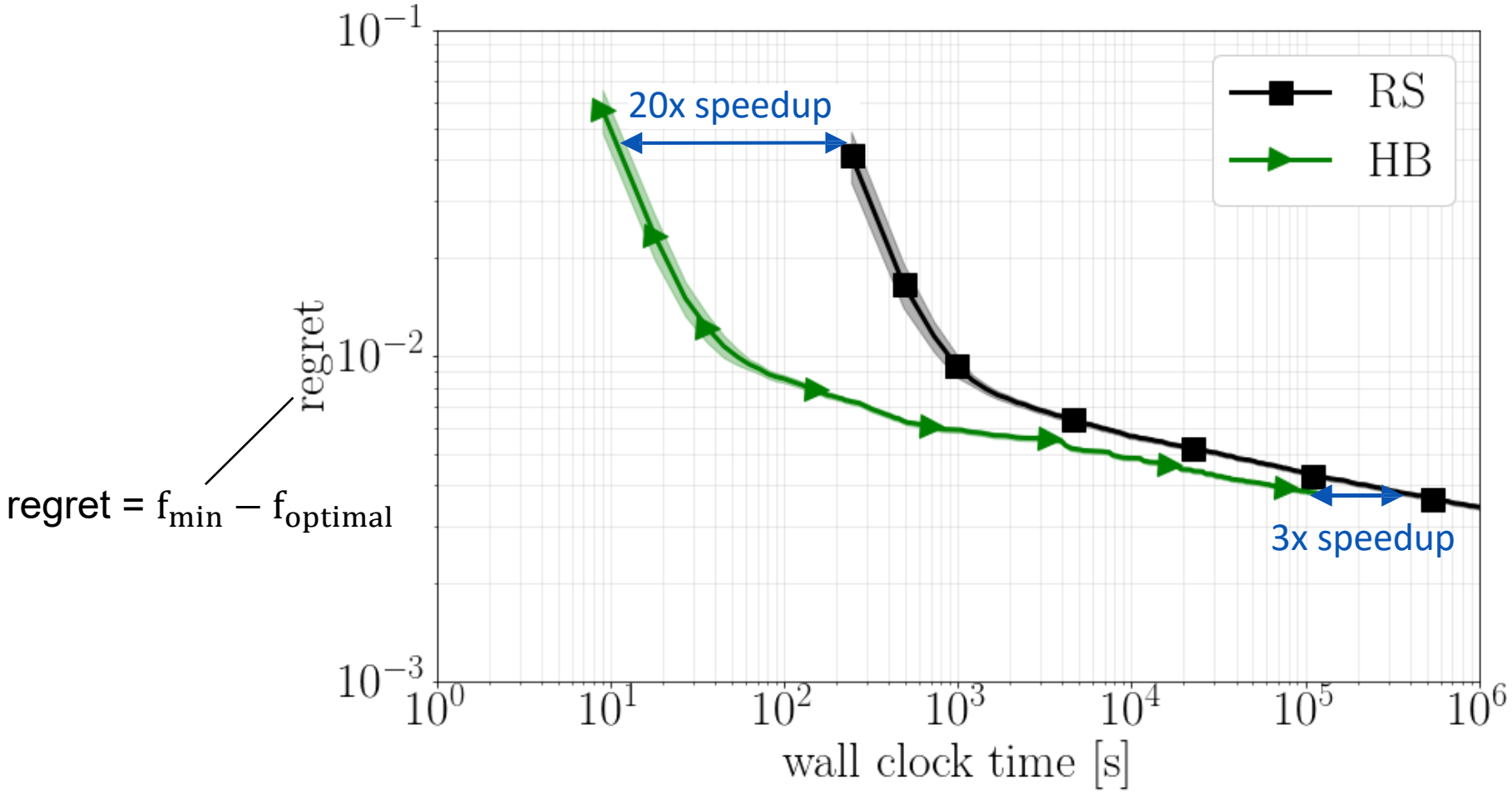
BOHB: Bayesian Optimization and HyperBand

- Advantage of Bayesian optimization
  - Strong final performance
- Advantage of HyperBand
  - Strong anytime performance
  - Easily parallelizable
  - Scalable

BOHB: Bayesian Optimization and HyperBand

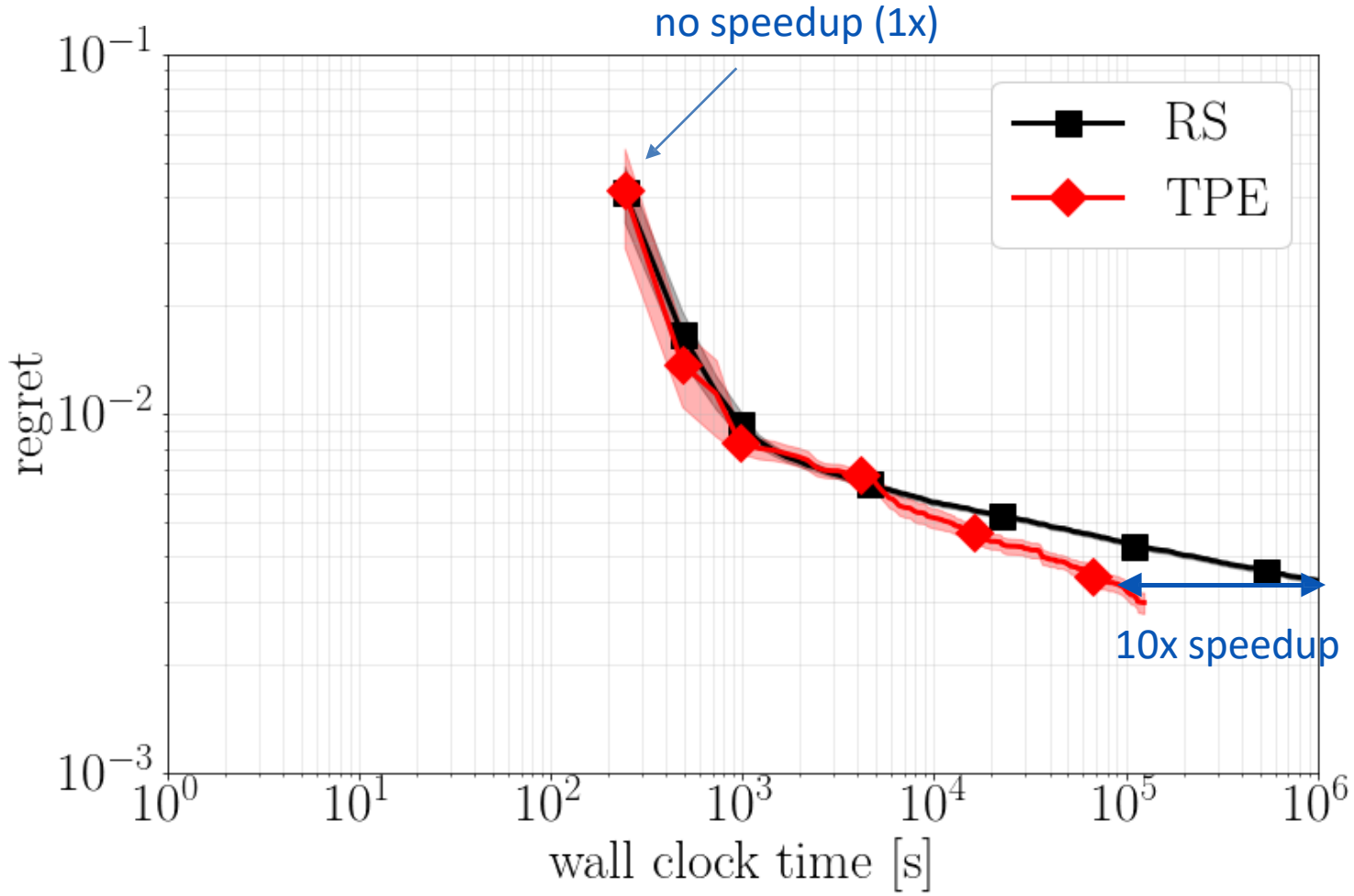
- Advantage of Bayesian optimization
  - Strong final performance
- Advantage of HyperBand
  - Strong anytime performance
  - Easily parallelizable
  - Scalable
- Combining the best of both: BOHB
  - Bayesian optimization (BO): **choosing the configuration** to evaluate (using TPE)
  - HyperBand (HB): deciding **how to allocate budgets**

HyperBand vs Random Search



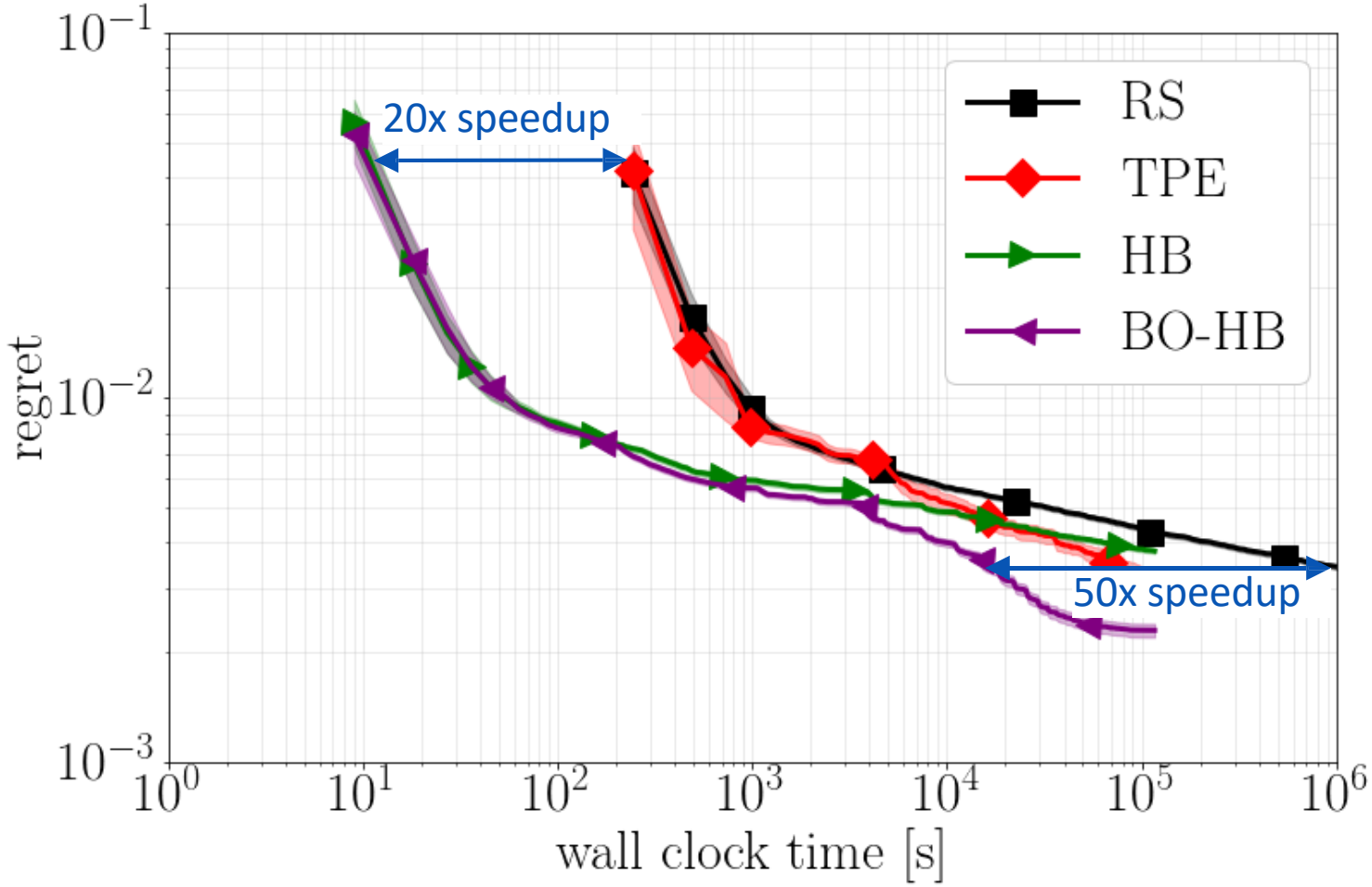
Biggest advantage: much improved **anytime performance**

Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Combining Bayesian Optimization & HyperBand



Best of both worlds: strong **anytime and final performance**

BOHB Algorithm

- Relies on HB to determine number of configurations and budget to evaluate
- Replaces the random selection of configurations at the beginning of each HB iteration by a TPE model-based search
- Update TPE using intermediate losses during SHA, prioritize results from higher budget
  - Use budget b = $argmax\{ D_b : |D_b| \geq N_{min} + 2\}$, where D_b is observation with budget b and N_{min} is minimum number of points to build a model

References

- AutoML tutorial at NeurIPS 2018[1)]
- AutoML-101[2)]
- Scalable Machine Learning and Deep Learning, KTH[3)]
- Automated machine learning[4)]

1) https://www.youtube.com/watch?v=0eBR8a4MQ30
2) https://www.automl.org/wp-content/uploads/2021/03/AutoML-101.pdf
3) https://id2223kth.github.io/slides/2022/11_automl.pdf
4) Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.