Speedup Techniques for Hyperparameter Optimization

BOHB

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Robust and Efficient Hyperparameter Optimization at Scale

Desiderata for a practical solution to the hyperparamenter optimization problem:

- Strong Anytime Performance
- Strong Final Performance
- Scalability
- Robustness & Flexibility
- Computational Efficiency
- Effective Use of Parallel Resources
- Conceptual / Algorithmic Simplicity

Robust and Efficient Hyperparameter Optimization at Scale

Desiderata for a practical solution to the hyperparamenter optimization problem:

- Strong Anytime Performance
- Strong Final Performance
- Scalability
- Robustness & Flexibility
- Computational Efficiency
- Effective Use of Parallel Resources
- Conceptual / Algorithmic Simplicity

To fulfill all of these desiderata, BOHB [Falkner, Klein and Hutter, ICML 2018] combines Bayesian Optimization with Hyperband

BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - Bayesian Optimization for choosing configurations to achieve strong final performance
  - Hyperband to choose the budgets for good anytime performance

BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - Bayesian Optimization for choosing configurations to achieve strong final performance
  - Hyperband to choose the budgets for good anytime performance

- BOHB replaces the random selection of configurations at the beginning of each HB iteration by a model-based search

BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - Bayesian Optimization for choosing configurations to achieve strong final performance
  - Hyperband to choose the budgets for good anytime performance
- BOHB replaces the random selection of configurations at the beginning of each HB iteration by a model-based search
- Details of the model
  - Variant of the Tree Parzen Estimator, with a product kernel
  - Models are fitted independently to the data for one budget at a time
    - Specifically, always the highest budget that has enough data points

BOHB: Algorithm

Pseudocode for sampling in BOHB

Input : observations D, fraction of random runs ρ, percentile γ, number of samples N_s, minimum number of points N_{min} to build a model, and bandwidth factor b_w

Output: next configuration to evaluate

if $rand() < \rho$ **then return** *random configuration*

$b = \arg\max\{D_b : |D_b| \geq N_{min} + 2\}$

if $b = \varnothing$ **then return** *random configuration*

Fit KDEs according to Equation **??**

Draw N_s samples acoording to $l'(\boldsymbol{\lambda})$

return *sample with highest ratio* $l(\boldsymbol{\lambda})/g(\boldsymbol{\lambda})$

BOHB: Algorithm

Pseudocode for sampling in BOHB

Input : observations D, fraction of random runs ρ, percentile γ, number of samples N_s, minimum number of points N_{min} to build a model, and bandwidth factor b_w

Output: next configuration to evaluate

if $rand() < \rho$ **then return** *random configuration*
$b = \arg\max\{D_b : |D_b| \geq N_{min} + 2\}$
if $b = \varnothing$ **then return** *random configuration*
Fit KDEs according to Equation **??**
Draw N_s samples acoording to $l'(\boldsymbol{\lambda})$
return *sample with highest ratio* $l(\boldsymbol{\lambda})/g(\boldsymbol{\lambda})$

$$l(\boldsymbol{\lambda}) = p(c(\boldsymbol{\lambda}) \leq \gamma | D_b) \qquad g(\boldsymbol{\lambda}) = p(c(\boldsymbol{\lambda}) > \gamma | D_b) \tag{1}$$

BOHB: Algorithm

Pseudocode for sampling in BOHB

Input : observations D, fraction of random runs ρ, percentile γ, number of samples N_s, minimum number of points N_{min} to build a model, and bandwidth factor b_w

Output: next configuration to evaluate

if $rand() < \rho$ **then return** *random configuration*
$b = \arg\max\{D_b : |D_b| \geq N_{min} + 2\}$
if $b = \varnothing$ **then return** *random configuration*
Fit KDEs according to Equation **??**
Draw N_s samples acoording to $l'(\boldsymbol{\lambda})$
return *sample with highest ratio* $l(\boldsymbol{\lambda})/g(\boldsymbol{\lambda})$

$$l(\boldsymbol{\lambda}) = p(c(\boldsymbol{\lambda}) \leq \gamma | D_b) \qquad g(\boldsymbol{\lambda}) = p(c(\boldsymbol{\lambda}) > \gamma | D_b) \tag{1}$$

- Note: $l'(\boldsymbol{\lambda})$ is similar to $l(\boldsymbol{\lambda},)$ but has larger bandwidths

BOHB: Empirical Evaluation

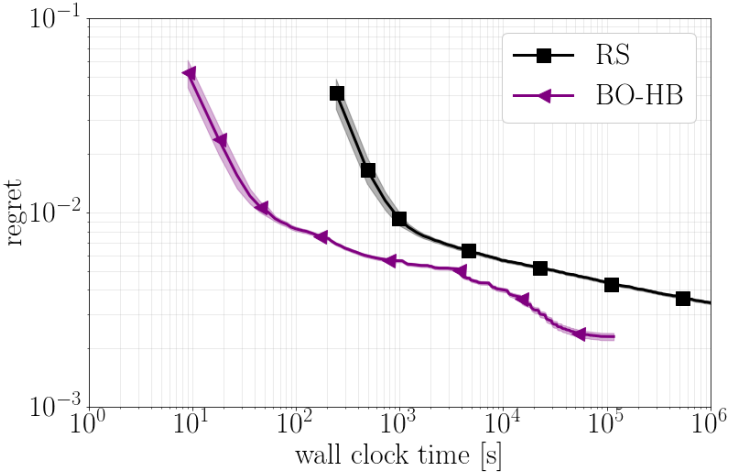

Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

BOHB: Empirical Evaluation

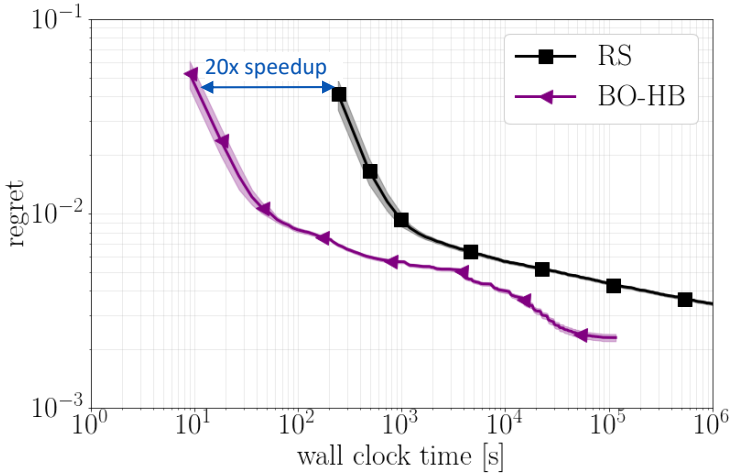

Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

BOHB: Empirical Evaluation

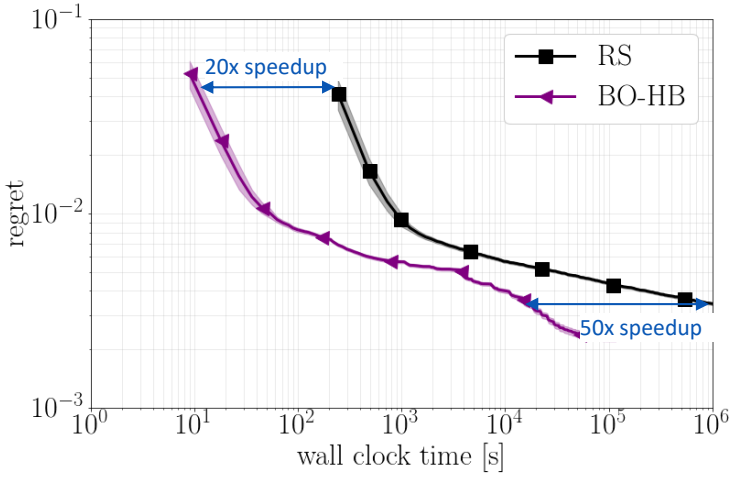

Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

BOHB: Empirical Evaluation

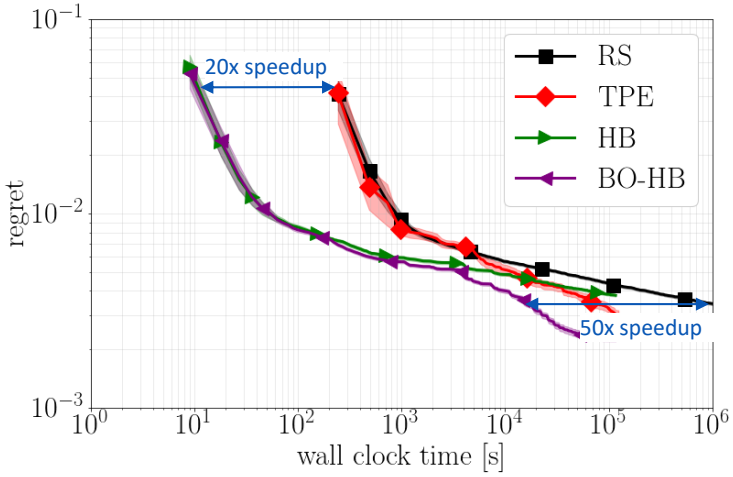

Figure: Performance of RS, TPE, HB and BOHB on Auto-Net on dataset Adult

BOHB: Different number of parallel workers

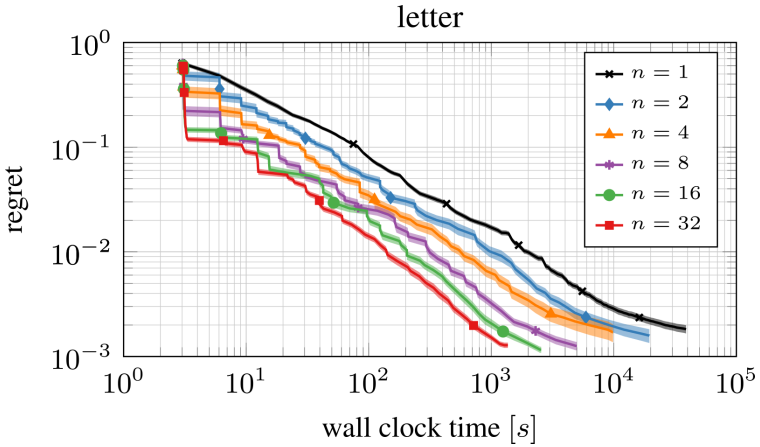

Figure: Performance of BOHB with different number of parallel workers on the letter surrogate benchmark for 128 iterations

BOHB: Optimization of a Bayesian Neural Network

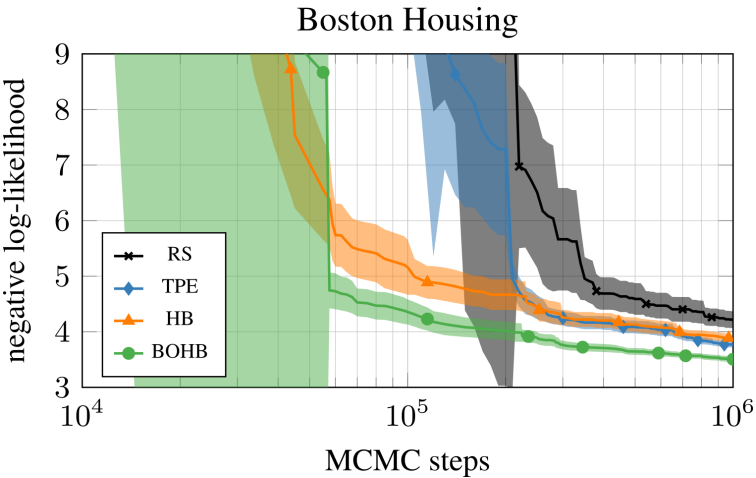

Figure: Optimization of 5 hyperparameters of a Bayesian neural network trained with SGHMC.

BOHB: Optimization of a Reinforcement Learning Agent

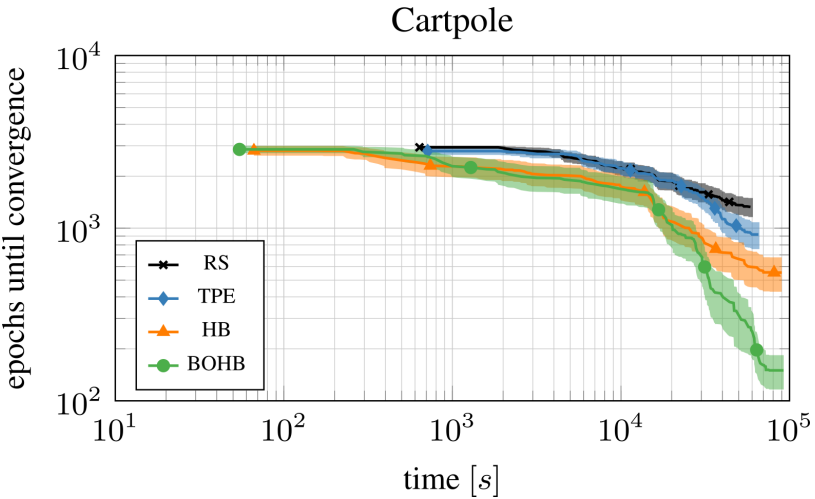

Figure: Hyperparameter optimization of 8 hyperparameters of PPO on the cartpole task.

BOHB: Optimization of an SVM

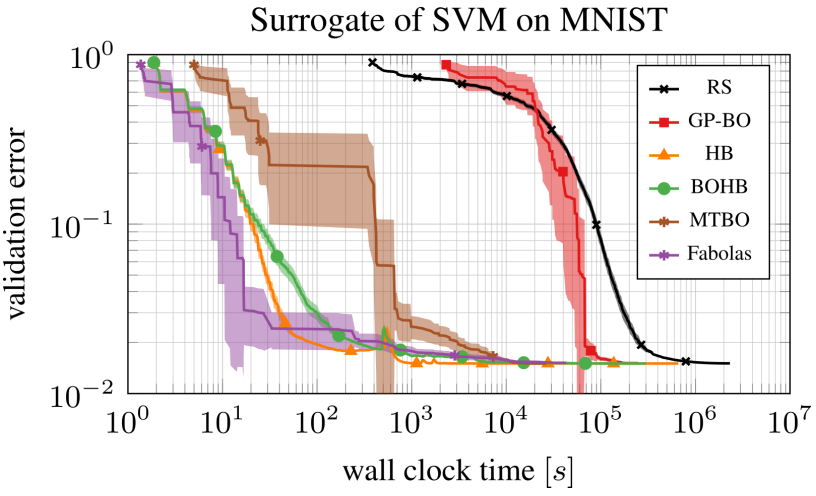

Figure: Comparison on the SVM on MNIST surrogate benchmark

BOHB: Counting Ones

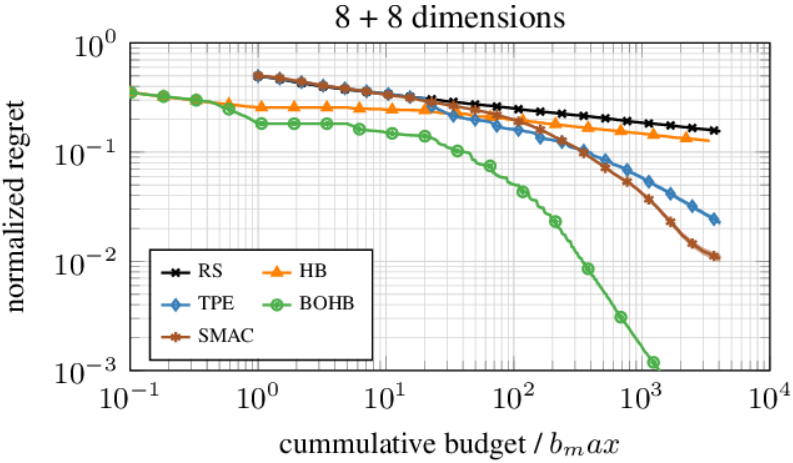

Figure: Results for the counting ones problem in 16 dimensional space with 8 categorical and 8 continuous hyperparameters.

BOHB: Optimization of a Feedforward Network

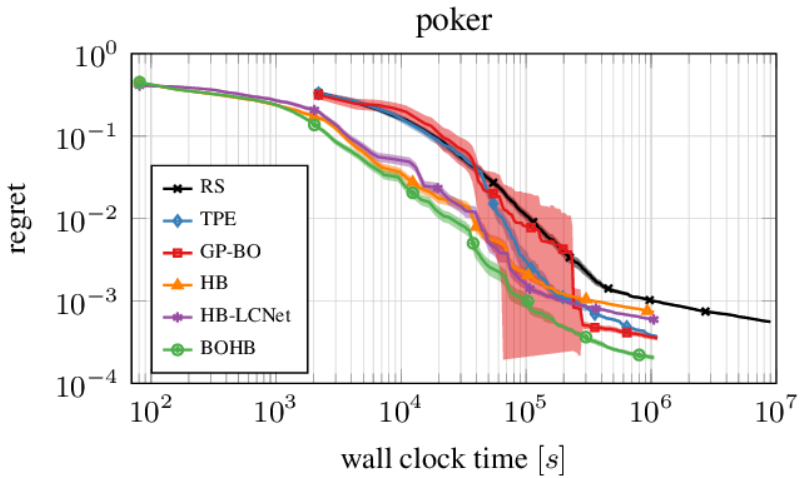

Figure: Optimizing six hyperparameters of a feed-forward neural network on featurized datasets; results are based on surrogate benchmarks.

Questions to Answer for Yourself / Discuss with Friends

- Repetition. Why does BOHB interleave randomly sampled configurations in the optimization process?
- Discussion. What are the advantages of the Parzen estimator model over more advanced models such as random forests or Gaussian processes?