BOHB: Robust and Efficient Hyperparameter Optimization at Scale

Stefan Falkner, Aaron Klein, Frank Hutter
{ sfalkner | kleinaa | fh }@cs.uni-freiburg.de
Department of Computer Science
University of Freiburg, Germany

UNI FREIBURG

Summary

- We propose and evaluate a hyperparameter optimizer that combines **B**ayesian **O**ptimization and **H**yper**B**and
- BOHB exploits low fidelity approximations and incorporates past evaluations into its model to speed up the optimization
- Our algorithms exhibits
  - strong anytime and final performance
  - efficient parallelization (multi-core machine or cluster)
  - scalability w.r.t. the search space dimensionality
  - flexibility towards different problem domains, i.e. continuous, discrete and mixed problems
  - robustness regarding different characteristics of the loss function, e.g., fidelity dependent noise or systematic differences across fidelities

Tree of Parzen Estimators (TPE)

- TPE (Berstra et al. 2011) is an instantiation of Bayesian Optimization
- Expected Improvement as the acquisition function

$$a(\boldsymbol{x}, \alpha) = \int \max(0, \alpha - f(\boldsymbol{x}))\, \mathrm{d}\, p(f(\boldsymbol{x})|D) \qquad (1)$$

- Non-parametric Parzen kernel density estimators (KDEs) to model the distribution of good and bad configurations w.r.t. a reference value α:

$$l(\boldsymbol{x}) = p(y < \alpha|\boldsymbol{x}) \qquad \text{and} \qquad g(\boldsymbol{x}) = p(y > \alpha|\boldsymbol{x}) \qquad (2)$$

- KDEs in (2) can be used to compute (1) and optimized via sampling
- TPE has been shown to scale to higher dimensions (Eggensperger et al. 2013) with little overhead and to parallelize easily (Berstra et al. 2011)

Hyperband (HB)

- HB (Li et al. 2017) iteratively allocates resources to random configurations using Successive-Halving (Jamieson and Talwalkar 2016).
- In each iteration HB selects N_i configurations for Successive-Halving which
  - runs many configurations on a small budget
  - increases the budget for the best ones
  - terminates a constant fraction at each step to limit the computational cost
- HB automatically trades off between simple random search (full budget) and a very aggressive early stopping (by evaluation on smaller budgets)
- HB is guaranteed to be at most a constant factor slower than random search
- If applicable, HB typically outperforms standard blackbox Bayesian optimization by exploiting cheap evaluations, e.g., subsets of the data, fewer iterations, limited execution time, or any continuous fidelity

BOHB

- BOHB takes advantage of smaller budgets (like HB) and previous evaluations (like TPE)
- model distributions for each budget of HB similar to TPE
  - TPE: hierarchy of one-dimensional KDEs
  - BOHB: single multidimensional KDE
- samples from a model replace random configurations
- small fraction of random configurations for guaranteed global convergence with at least the same rate as random search
- parallelization through limited optimization of the acquisition function to introduce diversity

Algorithm 1: BOHB's sampling procedure

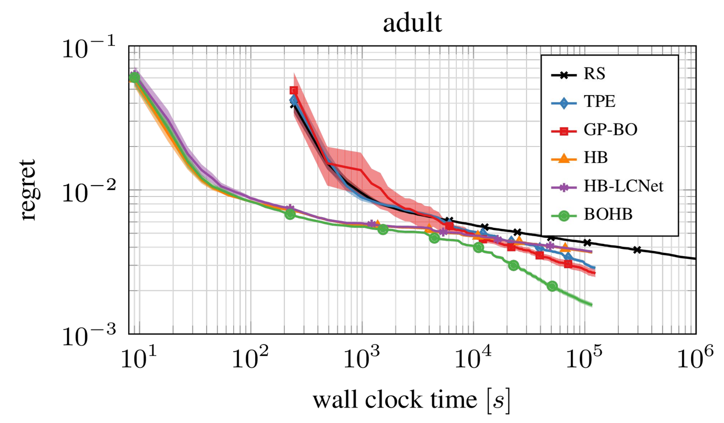
input : observations D, fraction of random runs ρ, percentile q, number of samples N_s, min number of points in a model N_{min}
output: next configuration to evaluate
if $rand() < \rho$ **then return** random configuration
find largest budget B with at least $N_{min}+1$ observations
if *no such* B *exists* **then return** random configuration
$\alpha = \mathrm{q}^{th}$ percentile of all y $\in D_b$
fit KDEs for probabilities in Eqs. (2)
draw N_s samples $\sim l(\boldsymbol{x})$
return sample with highest ratio $l(\boldsymbol{x})/g(\boldsymbol{x})$

Available under github.com/automl/HpBandSter

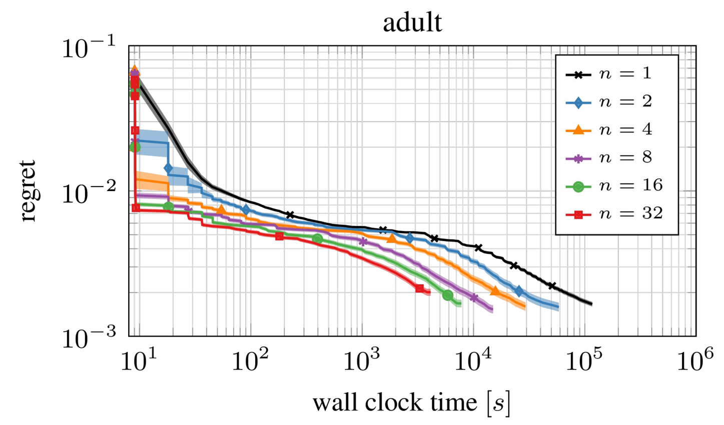
Experiments

- Benchmarks to evaluate performance:
  - Architecture and hyperparameters (6 parameters in total) of **Feed Forward Networks** on featurized data from OpenML (Vanschoren et al. 2014): Adult, Higgs, OptDigits, Letter, and Poker
    - To afford more runs, we build a surrogate (Eggensperger et al. 2015) based on 10000 random configurations each
    - Budget: training time
  - **Support Vector Machine** on MNIST (also a surrogate)
    - additional baselines: MTBO (Swersky et al. 2013), Fabolas (Klein et al. 2017); two competitive multi-fidelity optimizers
    - Budget: data subset size
  - **Proximal Policy Optimization** (Schulman et al. 2017) on OpenAI Gym (Brockman et al. 2016) environment cartpole
    - Budget: Number of independent trials
  - **Bayesian Neural Networks** via SGHMC (Chen et al. 2014) with scale adaptation (Springenberg et al 2016)
    - Budget: MCMC steps
  - A **Synthetic function** (a generalized counting ones) with arbitrary dimensionality (see paper for details)
    - additional base line: SMAC (Hutter et al. 2013)
    - Budget: draws from independent Bernoulli distributions, effectively controlling the noise
- Results:
  - Plots show average over 512 runs for FFNs, SVM and the synthetic function, and 50 runs for BNNs and PPO
  - Bayesian Optimization (TPE, GP-BO, SMAC) outperforms Random Search (RS) after about 30 function evaluations
  - TPE is similar to Random Search (RS) for the first ~30 evaluations, but better afterwards
  - HB and BOHB (and MTBO and Fabolas on the SVM) have strong performance early on by exploiting small budgets
  - Bayesian Optimization (TPE, GP-BO, SMAC) often outperforms HB for large optimization budgets but (usually) not BOHB

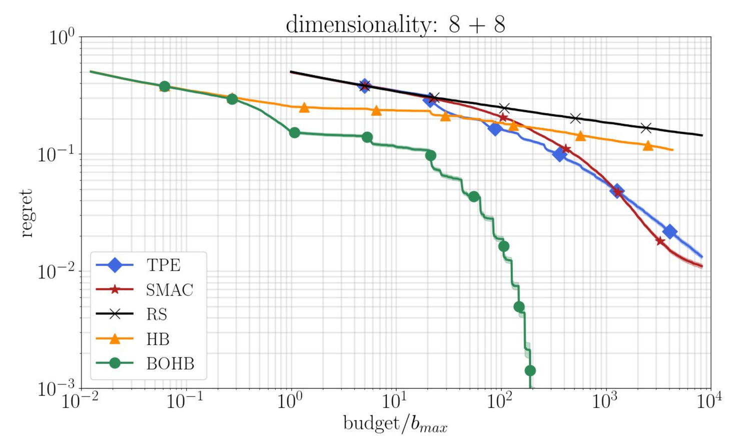
Feed Forward Networks



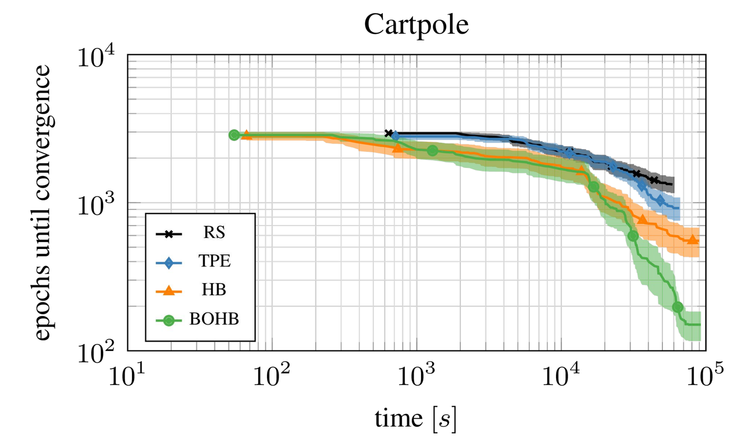
Evaluation with different numbers of parallel workers



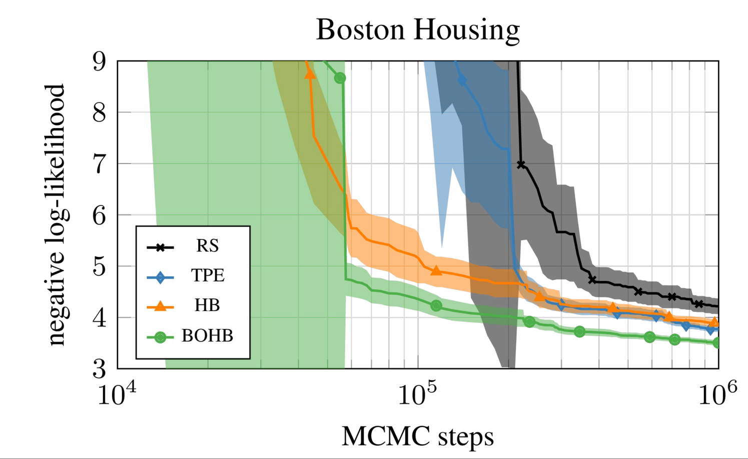
Synthetic function



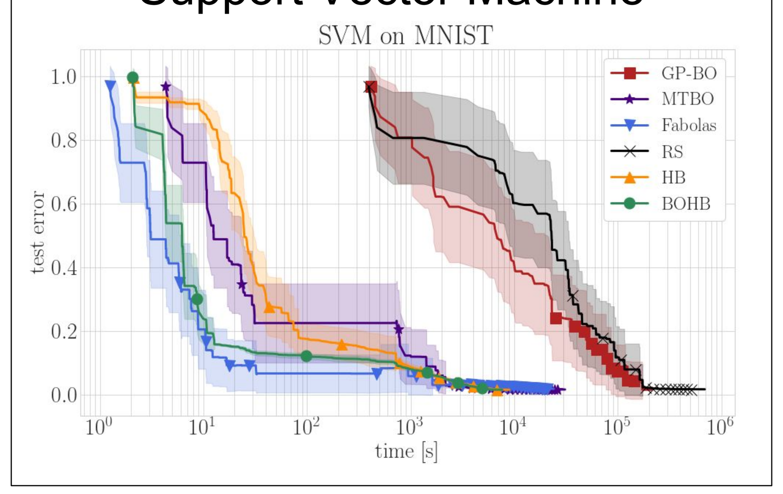
Proximal Policy Optimization



Bayesian Neural Networks



Support Vector Machine



AutoML.org