Meta-Surrogate Benchmarking for Hyperparameter Optimization

Aaron Klein*[1], Zhenwen Dai[2], Frank Hutter[3], Neil Lawrence[4], Javier Gonzalez[1]

[1]Amazon, [2]Spotify, [3]University of Freiburg, [4]University of Cambridge

*corresponding author: kleiaaro@amazon.com

In a nutshell

Benchmarking is a key step to fuel progress in an empirically driven field such as machine learning.
For hyperparameter optimization (HPO), benchmarking is particularly hard:

- most benchmarks are expensive, which renders extensive HPO computationally infeasible
- useful datasets are often scarce

We present Profet, a generative meta-model that allows to sample new HPO tasks. Thereby, tasks are returned in a parametric form and, hence, cheap-to-evaluate, which allows for exhaustive HPO benchmarking.

Dataset and code available at: https://github.com/amzn/emukit

Approach

Let $t_i \in \{t_0, \dots t_M\}$ be a set of related tasks with the same input domain $\mathcal{X}$ sampled from some distribution $p(t)$
Let us denote by $r(\alpha, t)$ the performance of an optimization method α on task t.
To draw statistically more significant conclusions, we would ideally like to integrate over all tasks:

$$S_{p(t)}(\alpha) = \int r(\alpha, t) p(t) dt,$$

Unfortunately, the above integral is intractable as $p(t)$ is unknown.
Profet approximates $p(t)$ with a generative meta-model $\hat{p}(t|\mathcal{D})$ based on some off-line generated data $\mathcal{D} = \{\{(\boldsymbol{x}_{tn}, y_{tn})\}_{n=1}^{N}\}_{t=1}^{T}$. This enables us to sample $t_i \sim \hat{p}(t \mid \mathcal{D})$ an arbitrary amount of tasks in order to perform a Monte-Carlo approximation:

$$S_p(\alpha) \approx \frac{1}{T} \sum_{i=1}^{T} r(a, t_i),$$

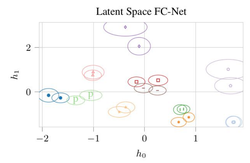
Meta-Model

Our Meta-Model for $\hat{p}(t|\mathcal{D})$ consists of two components:
- A probabilistic encoder $p(\mathbf{h}_t \mid \mathcal{D})$ based on GPLVM to model the correlation across tasks
- A multi task model: $p(y_t \mid \boldsymbol{x}, \mathbf{h}_t, \mathcal{D}) = \int p(y_t \mid \boldsymbol{x}, \mathbf{h}_t, \boldsymbol{\theta}) p(\theta \mid \mathcal{D}) d\boldsymbol{\theta} \approx \frac{1}{M} \sum_{i=1}^{M} p(y_t \mid \boldsymbol{x}, \mathbf{h}_t, \boldsymbol{\theta}_i).$

  based on a Bayesian neural networks. We assume $p(y_t \mid \boldsymbol{x}, \mathbf{h}_t) = \mathcal{N}(\mu(\boldsymbol{x}, \mathbf{h}_t), \sigma^2(\boldsymbol{x}, \mathbf{h}_t))$ to be Gaussian

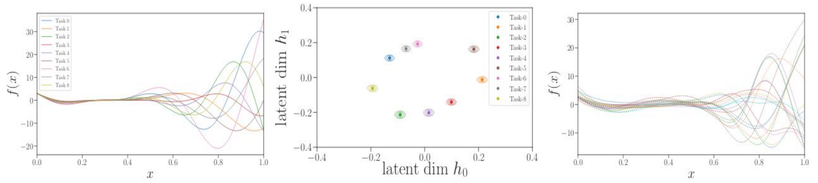
To generate new tasks $t_\star \sim \hat{p}(t \mid \mathcal{D})$ in a parametric form from our meta model:
1. Sample a new latent task vector $\mathbf{h}_{t_\star} \sim q(\mathbf{h}_t)$
2. Randomly samples a set of weight $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta} \mid \mathcal{D})$ from our Bayesian neural network
3. Set the function $f_{t_\star}(\boldsymbol{x}) = \hat{\mu}(\boldsymbol{x}, \mathbf{h}_{t_\star} \mid \boldsymbol{\theta}_i)$
4. Optionally we can emulate the observation noise by $y_{t_\star}(\boldsymbol{x}) \sim \mathcal{N}(\hat{\mu}(\boldsymbol{x}, \mathbf{h}_{t_\star} \mid \boldsymbol{\theta}_i), \hat{\sigma}^2_{\boldsymbol{\theta}_i})$

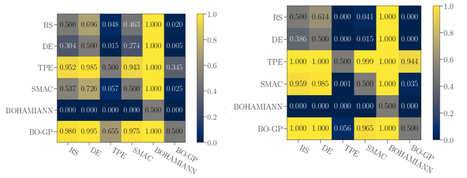
Evaluating the Meta-Model



Left: Representation (mean and 4 standard deviation) of task pairs (same color) generated by partitioning 11 datasets from the fully connected network benchmark (see below).

Right: Visualizing the concept of Profet: *Left:* 9 different tasks coming from the same distribution. *Middle:* latent space for the task embedding. *Right:* new tasks generated by our meta-model



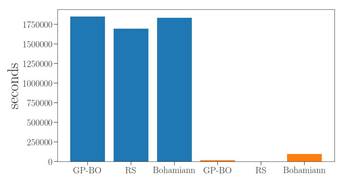


Left: p-values of pairwise comparisons between different HPO methods based on (*left*) 1000 real tasks, (*right)* results with 1000 tasks generated from our meta-model. Small p-values should be interpreted as finding evidence that the method in the column outperforms the method in the row.

Automated Benchmarking

We collected data for 3 typical HPO problem classes:
- support vector machine on OpenML classification datasets (D=2)
- Fully connected neural network on OpenML classification datasets (D=6)
- XGBoost on UCI regression datasets (D=8)

The plot on the right shows the computation time to perform 20 runs with different methods on real HPO tasks vs on surrogate tasks.



For each problem class we trained our meta-model and generated 1000 tasks. To showcase our benchmarking toolkit we evaluate various state-of-the-art Bayesian optimization methods, random search and two evolutionary algorithms. To aggregate performance across all tasks we report:
- empirical cumulative distribution (ecdf) of the runtime a optimizer requires to achieve a certain value
- average ranking scores

The plots below show the results exemplarily for the meta-svm tasks without noise