Surrogate Benchmarks for Hyperparameter Optimization

Katharina Eggensperger
Frank Hutter
University of Freiburg
{eggenspk,fh}@cs.uni-freiburg.de

Holger Hoos
Kevin Leyton-Brown
University of British Columbia
{hoos,kevinlb}@cs.ubc.ca
Problem:
Evaluation of Methods for Hyperparameter Optimization is expensive!
Outline

- Benchmarking Hyperparameter Optimization Methods
- Constructing Surrogates
- Using Surrogate Benchmarks
Outline

- Benchmarking Hyperparameter Optimization Methods
- Constructing Surrogates
- Using Surrogate Benchmarks
Bayesian Optimization Methods

- Surrogates for Hyperparameter Optimization Benchmarks – Eggensperger, Hutter, Hoos, and Leyton-Brown

Diagram:
- **Optimizer**
  - Uses internal model $M$
- **Configuration space $\Lambda$**
- **Performance** $f(\lambda_i)$
- **Run algorithm with configuration $\lambda_i$**

MetaseL’14
What do we need for an empirical comparison

- Standard benchmark problems
- Easy-to-use software

Then:
- Run each optimizer on each benchmark X multiple times
What do we need for an empirical comparison

- Standard benchmark problems
- Easy-to-use software

Then:
- Run each optimizer on each benchmark X multiple times

Evaluation of X is expensive
Benchmarking hyperparameter optimization methods

Neural Network, configuration space Λ:

- `anneal_start`
- `#hidden neurons`
- `learning rate`
- `batch_size`
- `iseed`
- `distribution`
- `squash`
- `scale heur.`
- `preproc.`
- `l2_penalty`
Benchmarking hyperparameter optimization methods

Neural Network, configuration space $\Lambda$: 

- anneal_start
- batch_size
- scale heur.
- #hidden neurons
- iseed
- preproc.
- learning rate
- distribution
- squash
- l2_penalty

categorical hyperparameter
Benchmarking hyperparameter optimization methods

Neural Network, configuration space $\Lambda$:

- **anneal_start**
- **#hidden_neurons**
- **learning_rate**
  - **batch_size**
  - **iseed**
  - **distribution**
  - **squash**

Conditional hyperparameter:

- **scale_heur = 0**
  - **scale_mult**
- **preproc = 1**
  - **colnorm_thresh**
- **preproc = 2**
  - **pca_energy**
- **l2_penalty = 1**
  - **l2_penalty_nz**
Benchmarks for hyperparameter optimization methods

Neural network

Best validation error achieved vs. #Function evaluations
Benchmarking hyperparameter optimization methods

Neural network

Best validation error achieved vs. #Function evaluations for different surrogate methods: SMAC_REAL, SPEARMINT_REAL, and TPE_REAL.
Outline

- Benchmarking Hyperparameter Optimization Methods
- Constructing Surrogates
- Using Surrogate Benchmarks
Surrogate Benchmark $X'$

- cheap-to-evaluate
- Can be used like the real benchmark $X$
- Behaves like $X$
Surrogate Benchmark $X'$

- cheap-to-evaluate
- Can be used like the real benchmark $X$
- Behaves like $X$

\[
\text{Regression model } X' \quad \text{Configuration } \lambda \quad \text{Performance } f(\lambda)
\]
Constructing a Surrogate for Benchmark X

1. Collect data
2. Choose a regression model
3. Train and store model
1. Collect data for benchmark X

Training data: \(((\lambda_1, f(\lambda_1)), \ldots, (\lambda_n, f(\lambda_n)))\)

- Dense sampling in high performance regions
- Good overall coverage
1. Collect data for benchmark X

Training data: \((\lambda_1, f(\lambda_1)), \ldots, (\lambda_n, f(\lambda_n))\)

- Dense sampling in high performance regions

Run optimizers on benchmark X

- Good overall coverage
1. Collect data for benchmark X

Training data: \(((\lambda_1, f(\lambda_1)), \ldots, (\lambda_n, f(\lambda_n)))\)

- Dense sampling in high performance regions

Run optimizers on benchmark X

- Good overall coverage

Run random search on benchmark X
2. Choice of Regression Models

<table>
<thead>
<tr>
<th>Ridge Regression</th>
<th>K-nearest neighbour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Boosting</td>
<td>Linear Regression</td>
</tr>
<tr>
<td>Random Forests</td>
<td>Gaussian Processes</td>
</tr>
<tr>
<td>Bayesian Neural Network</td>
<td>SVM</td>
</tr>
</tbody>
</table>
2. Choice of Regression Models

<table>
<thead>
<tr>
<th>Ridge Regression</th>
<th>K-nearest neighbour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Boosting</td>
<td>Linear Regression</td>
</tr>
<tr>
<td>Random Forests</td>
<td>Gaussian Processes</td>
</tr>
<tr>
<td>Bayesian Neural Network</td>
<td>SVM</td>
</tr>
</tbody>
</table>
2. Choice of Regression Models

Can we quantify the performance of a new optimizer?

- **Leave-one-optimizer-out** setting
  - Train model on data gathered by all but one optimizer
  - Test on remaining data

Surrogates for Hyperparameter Optimization Benchmarks – Eggensperger, Hutter, Hoos, and Leyton-Brown
2. Choice of Regression Models

Leave-one-optimizer-out setting

![Graph showing comparison between Random forest prediction and Neural Network against True performance.](image)
2. Choice of Regression Models
Leave-one-optimizer-out setting

Random Forest

Neural Network
2. Choice of Regression Models
Leave-one-optimizer-out setting

Random Forest
Gaussian Process
k-nearest-neighbour

Gradient Boosting
nuSVR

Neural Network
## 2. Choice of Regression Models

<table>
<thead>
<tr>
<th>Ridge Regression</th>
<th>K-nearest neighbour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Boosting</td>
<td>Linear Regression</td>
</tr>
<tr>
<td><strong>Random Forests</strong></td>
<td><strong>Gaussian Processes</strong></td>
</tr>
<tr>
<td>Bayesian Neural Network</td>
<td>SVM</td>
</tr>
</tbody>
</table>
Outline

- Benchmarking Hyperparameter Optimization Methods
- Constructing Surrogates
- Using Surrogate Benchmarks
Using Surrogate Benchmarks

Neural Network

Best validation error achieved against #Function evaluations for different surrogate benchmarks: SMAC_REAL, SPEARMINT_REAL, and TPE_REAL.
Using Surrogate Benchmarks

Neural Network

Real Benchmark
Using Surrogate Benchmarks

Neural Network

Real Benchmark
RF-based benchmark
GP-based benchmark
Using Surrogate Benchmarks

Neural Network

Real Benchmark

RF-based benchmark

GP-based benchmark

One optimization run: 40h <200s <200s

Whole comparison: 50d <1.5h <1.5h
Applications

- Extensive testing at **early development stages**
- **Fast comparison** of different hyperparameter optimization methods
- **Metaoptimization** of existing hyperparameter optimization methods
Conclusion

Can we construct cheap-to evaluate and realistic hyperparameter optimization benchmarks?

Yes, based on random forests and Gaussian process regression models
Conclusion

Can we construct cheap-to evaluate and realistic hyperparameter optimization benchmarks?

Yes, based on random forests and Gaussian process regression models

But, some work needs to be done for high dimensional benchmarks.
This presentation was supported by an *ECCAI travel award* and the *ECCAI sponsors*

Thank you for your attention

More information on hyperparameter optimization benchmarks can be found on [automl.org/hpolib](https://automl.org/hpolib)
## Regression models

<table>
<thead>
<tr>
<th>Model</th>
<th>Hyperparameter optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest (RF)</td>
<td>None</td>
</tr>
<tr>
<td>Gradient Boosting (GB)</td>
<td>None</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>None</td>
</tr>
<tr>
<td>Gaussian process (GP), Matérn 5/2 kernel</td>
<td>MCMC sampling over hyperparameters</td>
</tr>
<tr>
<td>Support Vector Regression (SVR)</td>
<td>Random search for C and gamma</td>
</tr>
<tr>
<td>NuSVR</td>
<td>Random search for C, gamma and nu</td>
</tr>
<tr>
<td>Bayesian neural network (BNN)</td>
<td>None</td>
</tr>
<tr>
<td>k-nearest neighbour (KNN)</td>
<td>Random search for n_neighbors</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>None</td>
</tr>
<tr>
<td>Least Angle Regression</td>
<td>None</td>
</tr>
<tr>
<td>Ridge Regression</td>
<td>None</td>
</tr>
</tbody>
</table>
### Benchmarks

<table>
<thead>
<tr>
<th></th>
<th>#(\lambda)</th>
<th>cond.</th>
<th>cat. / cont.</th>
<th>Input dim.</th>
<th>#evals. per run</th>
<th>#data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branin</td>
<td>2</td>
<td>-</td>
<td>- / 2</td>
<td>3</td>
<td>200</td>
<td>7402</td>
</tr>
<tr>
<td>Log. Reg. 5CV</td>
<td>4</td>
<td>-</td>
<td>- / 4</td>
<td>9</td>
<td>500</td>
<td>18521</td>
</tr>
<tr>
<td>HP-NNET convex</td>
<td>14</td>
<td>4</td>
<td>7 / 7</td>
<td>25</td>
<td>200</td>
<td>7750</td>
</tr>
<tr>
<td>HP-DBNET mrbi</td>
<td>36</td>
<td>27</td>
<td>19 / 17</td>
<td>82</td>
<td>200</td>
<td>7466</td>
</tr>
</tbody>
</table>