Benchmarking Beyond Branin

Katharina Eggensperger
University of Freiburg, Germany
The case for solid benchmarks

Progress in classification of images. The error rate (%) of the ImageNet competition winner by year compared to a human annotator (red line).
The case for solid benchmarks

- Enable & track progress of a research community
- Identify strengths and weaknesses of algorithms
- Compare different methods
### HPOlib: Once upon a time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>#hyp.params (conditional)</th>
<th>continuous/discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branin</td>
<td>2(-)</td>
<td>2/-</td>
</tr>
<tr>
<td>Hartmann 6d</td>
<td>6(-)</td>
<td>6/-</td>
</tr>
<tr>
<td>Log. Reg.</td>
<td>4(-)</td>
<td>4/-</td>
</tr>
<tr>
<td>LDA ongrid</td>
<td>3(-)</td>
<td>-/3</td>
</tr>
<tr>
<td>SVM ongrid</td>
<td>3(-)</td>
<td>-/3</td>
</tr>
<tr>
<td>HP-NNET</td>
<td>14(4)</td>
<td>7/7</td>
</tr>
<tr>
<td>HP-NNET 5CV</td>
<td>14(4)</td>
<td>7/7</td>
</tr>
<tr>
<td>HP-DBNET</td>
<td>36(27)</td>
<td>19/17</td>
</tr>
<tr>
<td>Auto-WEKA</td>
<td>786(784)</td>
<td>296/490</td>
</tr>
<tr>
<td>Log. Reg. 5CV</td>
<td>4(-)</td>
<td>4/-</td>
</tr>
<tr>
<td>HP-NNET 5CV</td>
<td>14(4)</td>
<td>7/7</td>
</tr>
<tr>
<td>HP-NNET 5CV</td>
<td>14(4)</td>
<td>7/7</td>
</tr>
</tbody>
</table>

Logistic Regression on MNIST
[Snoek et al., 2012; LeCun et al., 1998]

4 continuous hyperparameters

>1.5h for 100 function evaluations

HPDBNET on MRBI
[Bergstra et al, 2011; Larochelle et al, 2007]

36 mixed continuous and discrete hyperparameters

>48h for 200 function evaluations on a GPU

Speed up your experiments

Configuration Space $\Lambda$

Select $\lambda \in \Lambda$ → Assess Performance $f(\lambda)$ → Best performing $\lambda^*$

Hyperparameter Optimization
Speed up your experiments

Configuration Space $\Lambda$

Select $\lambda \in \Lambda$ → Surrogate predicts $f(\lambda)$ → Best performing $\lambda^*$

Hyperparameter Optimization
How to construct the surrogate?

1. Collect $\langle \lambda, f(\lambda) \rangle$ pairs that
   - cover configuration space with a focus on high-performing regions

2. Fit a regression model $\hat{f}$ that
   - scales to large datasets
   - does fast predictions
   - has a high accuracy
How to construct the surrogate?

1. **Collect** $< \lambda, f(\lambda) >$ pairs that
   - cover configuration space with a focus on high-performing regions
   → Reuse data collected during hyperparameter optimization

2. **Fit a regression model** $\hat{f}$ that
   - scales to large datasets
   - does fast predictions
   - has a high accuracy
   → (Approximate) GPs, kNN, SVRs, Neural Networks, Random Forests, …
Logistic Regression on MNIST
[Snoek et al, 2012; Y. LeCun et al, 1998]

- 4 continuous hyperparameters
Surrogate Benchmarks

Logistic Regression on MNIST
[Snoek et al, 2012; Y. LeCun et al, 1998]

- 4 continuous hyperparameters

Eggersperger, Hutter, Hoos, Leyton-Brown: "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates", AAAI'15
Surrogate Benchmarks

**HP-DBNET on MRBI**

[Bergstra et al 2011; Larochelle et al, 2007]

- 36 mixed hyperparameter

Eggersperger, Hutter, Hoos, Leyton-Brown: "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates", AAAI'15
Surrogate Benchmarks

HP-DBNET on MRBI
[Bergstra et al 2011; Larochelle et al, 2007]

- 36 mixed hyperparameter

True function evaluation: 15min
Surrogate benchmark prediction: <1 sec

Eggersperger, Hutter, Hoos, Leyton-Brown: "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates", AAAI'15
HPOlib: Summary

- Every optimization method has strengths and weaknesses

- Surrogate-based benchmark problems allow for:
  - easy debugging
  - efficient comparisons

<table>
<thead>
<tr>
<th>Artificial Functions</th>
<th>Hyperparameter Optimization Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realistic</td>
<td>✔</td>
</tr>
<tr>
<td>easy to set up</td>
<td>✔</td>
</tr>
<tr>
<td>cheap to run</td>
<td>✔</td>
</tr>
<tr>
<td>runnable w/o special hardware</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
HPOlib: Summary

- Every optimization method has strengths and weaknesses

- Surrogate-based benchmark problems allow for:
  - easy debugging
  - efficient comparisons

<table>
<thead>
<tr>
<th>Artificial Functions</th>
<th>HPOlib</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realistic</td>
<td>✗</td>
</tr>
<tr>
<td>easy to set up</td>
<td>✓</td>
</tr>
<tr>
<td>cheap to run</td>
<td>✓</td>
</tr>
<tr>
<td>runnable w/o special hardware</td>
<td>✓</td>
</tr>
</tbody>
</table>
HPOlib: Summary

- Every optimization method has strengths and weaknesses

- Surrogate-based benchmark problems allow for:
  - easy debugging
  - efficient comparisons

<table>
<thead>
<tr>
<th>Artificial Functions</th>
<th>Surrogate benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realistic</td>
<td>✗</td>
</tr>
<tr>
<td>easy to set up</td>
<td>✓</td>
</tr>
<tr>
<td>cheap to run</td>
<td>✓</td>
</tr>
<tr>
<td>runnable w/o special hardware</td>
<td>✓</td>
</tr>
</tbody>
</table>
Going Beyond

- Branin
- cost
- tasks

complexity of hyperparameter space
Going Beyond

cost

tasks

complexity of hyperparameter space
Complex Hyperparameter Spaces

- Challenges:
  - Many dimensions
  - Mixed categorical and continuous hyperparameters
  - Conditional hyperparameters

- Examples:
  - Auto-Weka [Thornton et al, 2013]
  - Auto-sklearn [Feurer et al, 2015]
Auto-sklearn’s pipeline

M. Feurer and A. Klein and K. Eggensperger and J. Springenberg and M. Blum and F. Hutter: "Efficient and Robust Automated Machine Learning", NIPS'15
Auto-sklearn’s Configuration Space

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-hot encoding</td>
<td>2</td>
</tr>
<tr>
<td>imputation</td>
<td>1</td>
</tr>
<tr>
<td>balancing</td>
<td>1</td>
</tr>
<tr>
<td>rescaling</td>
<td>1</td>
</tr>
</tbody>
</table>
### Auto-sklearn’s Configuration Space

<table>
<thead>
<tr>
<th>name</th>
<th>#(\lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>extreml. rand. trees prep.</td>
<td>5</td>
</tr>
<tr>
<td>fast ICA</td>
<td>4</td>
</tr>
<tr>
<td>feature agglomeration</td>
<td>4</td>
</tr>
<tr>
<td>kernel PCA</td>
<td>5</td>
</tr>
<tr>
<td>rand. kitchen sinks</td>
<td>2</td>
</tr>
<tr>
<td>linear SVM prep.</td>
<td>3</td>
</tr>
<tr>
<td>no preprocessing</td>
<td>-</td>
</tr>
<tr>
<td>nystroem sampler</td>
<td>5</td>
</tr>
<tr>
<td>PCA</td>
<td>2</td>
</tr>
<tr>
<td>polynomial</td>
<td>3</td>
</tr>
<tr>
<td>random trees embed.</td>
<td>4</td>
</tr>
<tr>
<td>select percentile</td>
<td>2</td>
</tr>
<tr>
<td>select rates</td>
<td>3</td>
</tr>
<tr>
<td>one-hot encoding</td>
<td>2</td>
</tr>
<tr>
<td>imputation</td>
<td>1</td>
</tr>
<tr>
<td>balancing</td>
<td>1</td>
</tr>
<tr>
<td>rescaling</td>
<td>1</td>
</tr>
</tbody>
</table>
### Auto-sklearn’s Configuration Space

<table>
<thead>
<tr>
<th>name</th>
<th>#(\lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>extreml. rand. trees prepr.</td>
<td>5</td>
</tr>
<tr>
<td>fast ICA</td>
<td>4</td>
</tr>
<tr>
<td>feature agglomeration</td>
<td>4</td>
</tr>
<tr>
<td>kernel PCA</td>
<td>5</td>
</tr>
<tr>
<td>rand. kitchen sinks</td>
<td>2</td>
</tr>
<tr>
<td>linear SVM prepr.</td>
<td>3</td>
</tr>
<tr>
<td>no preprocessing</td>
<td>-</td>
</tr>
<tr>
<td>nystroem sampler</td>
<td>5</td>
</tr>
<tr>
<td>PCA</td>
<td>2</td>
</tr>
<tr>
<td>polynomial</td>
<td>3</td>
</tr>
<tr>
<td>random trees embed.</td>
<td>4</td>
</tr>
<tr>
<td>select percentile</td>
<td>2</td>
</tr>
<tr>
<td>select rates</td>
<td>3</td>
</tr>
<tr>
<td>one-hot encoding</td>
<td>2</td>
</tr>
<tr>
<td>imputation</td>
<td>1</td>
</tr>
<tr>
<td>balancing</td>
<td>1</td>
</tr>
<tr>
<td>rescaling</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>#(\lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost (AB)</td>
<td>4</td>
</tr>
<tr>
<td>Bernoulli naïve Bayes</td>
<td>2</td>
</tr>
<tr>
<td>decision tree (DT)</td>
<td>4</td>
</tr>
<tr>
<td>extreml. rand. trees</td>
<td>5</td>
</tr>
<tr>
<td>Gaussian naïve Bayes</td>
<td>-</td>
</tr>
<tr>
<td>gradient boosting (GB)</td>
<td>6</td>
</tr>
<tr>
<td>kNN</td>
<td>3</td>
</tr>
<tr>
<td>LDA</td>
<td>4</td>
</tr>
<tr>
<td>linear SVM</td>
<td>4</td>
</tr>
<tr>
<td>kernel SVM</td>
<td>7</td>
</tr>
<tr>
<td>multinomial naïve Bayes</td>
<td>2</td>
</tr>
<tr>
<td>passive aggressive</td>
<td>3</td>
</tr>
<tr>
<td>QDA</td>
<td>2</td>
</tr>
<tr>
<td>random forest (RF)</td>
<td>5</td>
</tr>
<tr>
<td>Linear Class. (SGD)</td>
<td>10</td>
</tr>
</tbody>
</table>
Auto-sklearn’s Configuration Space

data preprocessor

rescaling
- min/max
- standard

one hot enc.

imputation
- mean
- median

balancing
- weighting
- None

Auto-sklearn’s Configuration Space

**Data Preprocessor**
- Rescaling: min/max, standard
- One hot encoding
- Imputation: mean, median
- Balancing: weighting, None

**Feature Preprocessor**
- Preprocessing: PCA, None, fast ICA

---

Auto-sklearn’s Configuration Space

Data preprocessor:
- rescaling: min/max, standard
- one hot enc.
- imputation: mean, median
- balancing: weighting, None

Feature preprocessor:
- PCA
- None
- fast ICA

Estimator:
- RF
- AdaBoost
- kNN

Learning rate, # estimators, max. depth

Bayesian Optimization can handle that space
Bayesian Optimization can handle that space
Auto-sklearn Workflow

https://github.com/automl/auto-sklearn

Auto-sklearn Workflow

https://github.com/automl/auto-sklearn

Auto-sklearn Workflow

https://github.com/automl/auto-sklearn

M. Feurer and A. Klein and K. Eggensperger and J. Springenberg and M. Blum and F. Hutter: "Efficient and Robust Automated Machine Learning", NIPS'15
Going Beyond

- Cost
- Tasks
- Complexity of hyperparameter space
Expensive Target Functions

Challenges:
- Expensive runs
- Only few function evaluations feasible

Examples:
- Robotics
- Deep Learning
Expensive Target Functions

Challenges:
- Expensive runs
- Only few function evaluations feasible

Examples:
- Robotics
- Deep Learning

Recent approaches:
- MTBO [Swersky et al, 2013]
- Freeze-Thaw Bayesian optimization [Swersky et al, 2014]
- Hyperband [Li et al, 2016]
- Multi-fidelity Gaussian Process Bandit Optimization [Kandasamy et al, 2016]
- FABOLAS [Klein et al, 2016]
Example: Fabolas

**Fast Bayesian Optimization on Large Data Sets**

https://github.com/automl/RoBO

Example: Fabolas

**FAst Baysian Optimization on LArge Data Sets**

https://github.com/automl/RoBO

A. Klein and S. Falkner and S. Bartels and P. Hennig and F. Hutter: "Fast Bayesian optimization of Machine Learning Hyperparameters on Large Datasets", ArXiv’16
Example: Fabolas

**FAst B**ayesian **O**ptimization on **LA**rge Data **S**ets

https://github.com/automl/RoBO

A. Klein and S. Falkner and S. Bartels and P. Hennig and F. Hutter: "Fast Bayesian optimization of Machine Learning Hyperparameters on Large Datasets", ArXiv'16
Example: Fabolas

**Fast Bayesian Optimization on Large Data Sets**

A. Klein and S. Falkner and S. Bartels and P. Hennig and F. Hutter: "Fast Bayesian optimization of Machine Learning Hyperparameters on Large Datasets", ArXiv'16

[https://github.com/automl/RoBO](https://github.com/automl/RoBO)
Example: Fabolas

**Fast Bayesian Optimization on Large Data Sets**

Small data subsets suffice to estimate performance of a configuration

→ Model data set size as an additional degree of freedom

https://github.com/automl/RoBO

A. Klein and S. Falkner and S. Bartels and P. Hennig and F. Hutter: "Fast Bayesian optimization of Machine Learning Hyperparameters on Large Datasets", ArXiv’16
Example: Fabolas

ResNet on CIFAR-10

[He et al, 2015]

- 4 continuous hyperparameters

Going Beyond

complexity of hyperparameter space
cost
tasks
Optimizing across tasks

- **Challenges:**
  - Tune hyperparameters across a set of tasks

\[ \lambda^* \in \arg\min_{\lambda \in \Lambda} \mathbb{E}_{t \in T} f_t (\lambda) \]

- **Examples:**
  - Hyperparameter tuning across crossvalidation folds
  - General algorithm configuration
Example: ProbSAT on 7Sat90
Example: ProbSAT on 7Sat90

→ Objective value varies across tasks
Example: ProbSAT on 7Sat90
Example: ProbSAT on 7Sat90

→ Widely varying running time distributions depending on seed
Algorithm Configuration

- Varying objective value across tasks
- Large noise
Algorithm Configuration

- Varying objective value across tasks
  → Reject parameter setting before evaluating it on all tasks

- Large noise
  → Evaluate runs multiple times

Bayesian Optimization with Random Forests can handle this data
## Available Benchmarks: AClib2

<table>
<thead>
<tr>
<th>Solver</th>
<th>Domain</th>
<th># Params</th>
<th># Instances</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clasp</td>
<td>ASP</td>
<td>90</td>
<td>240/240</td>
<td>4d</td>
</tr>
<tr>
<td>CPLEX</td>
<td>MIP</td>
<td>73</td>
<td>1000/1000</td>
<td>2d</td>
</tr>
<tr>
<td>LPG</td>
<td>Planning</td>
<td>67</td>
<td>2000/2000</td>
<td>2d</td>
</tr>
<tr>
<td>ProbSat</td>
<td>SAT</td>
<td>9</td>
<td>250/250</td>
<td>3h</td>
</tr>
<tr>
<td>Xgboost</td>
<td>ML</td>
<td>11</td>
<td>10/1</td>
<td>500 runs</td>
</tr>
<tr>
<td>SVM</td>
<td>ML</td>
<td>7</td>
<td>10/1</td>
<td>500 runs</td>
</tr>
</tbody>
</table>

https://bitbucket.org/mlindauer/aclib2
Can we build surrogate models for these benchmark problems, too?
How to construct the surrogate?

1. **Collect** $< \lambda, t, f(\lambda, t) >$ tuples that
   - cover configuration space with a focus on high-performing regions

2. **Fit a regression model** $\hat{f}$ that
   - scales to large datasets & does fast predictions & has a high accuracy
   - Mimics distribution of the objective value
How to construct the surrogate?

1. **Collect** \(< \lambda, t, f(\lambda, t) >\) **tuples that**
   - cover configuration space with a focus on **high-performing regions**
   → **Reuse data collected during configuration**

2. **Fit a regression model** \(\hat{f}\) **that**
   - scales to **large datasets** & does fast predictions & has a **high accuracy**
   - Mimics distribution of the objective value
   → **Quantile Regression Forests**
Can we build surrogate models for these benchmark problems, too?

→ Run your experiments more than 100 times faster!

https://bitbucket.org/mlindauer/aclib2 on branch Surrogates
Introducing HPOlib2

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>#hyper-parameter</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Functions</td>
<td>1-X</td>
<td>-</td>
</tr>
<tr>
<td>Auto-sklearn</td>
<td>&gt;100</td>
<td>OpenML</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>4</td>
<td>MNIST</td>
</tr>
<tr>
<td>SVM</td>
<td>2</td>
<td>MNIST</td>
</tr>
<tr>
<td>ResNet</td>
<td>4</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>ConvNet</td>
<td>5</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>Fully Connected Net</td>
<td>10</td>
<td>MNIST</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[https://github.com/automl/HPOlib2](https://github.com/automl/HPOlib2)
HPOlib2: easy-to-use

```python
from hpolib.benchmarks.ml import svm_benchmark
```
HPOlib2: easy-to-use

```python
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()
```
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()

# Evaluate one configuration
b.objective_function(configuration=[5, -5])
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()

# Evaluate one configuration
b.objective_function(configuration=[5, -5])

# Returns running time and loss
# {'cost': 251.88, 'function_value': 0.012}
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()

# Evaluate one configuration on subset
b.objective_function(configuration=[5, -5],
                       dataset_fraction=0.5)

# Returns running time and loss
# {'cost': 110.80, 'function_value': 0.025}
Summary & Conclusion

- **Solid benchmark problems** (beyond Branin)
  - track progress
  - thorough comparisons
  - reproducible research

- **Surrogate-based benchmarks**
  - rapid development
  - feasible large-scale experiments

https://bitbucket.org/mlindauer/aclib2
https://github.com/automl/HPOlib2