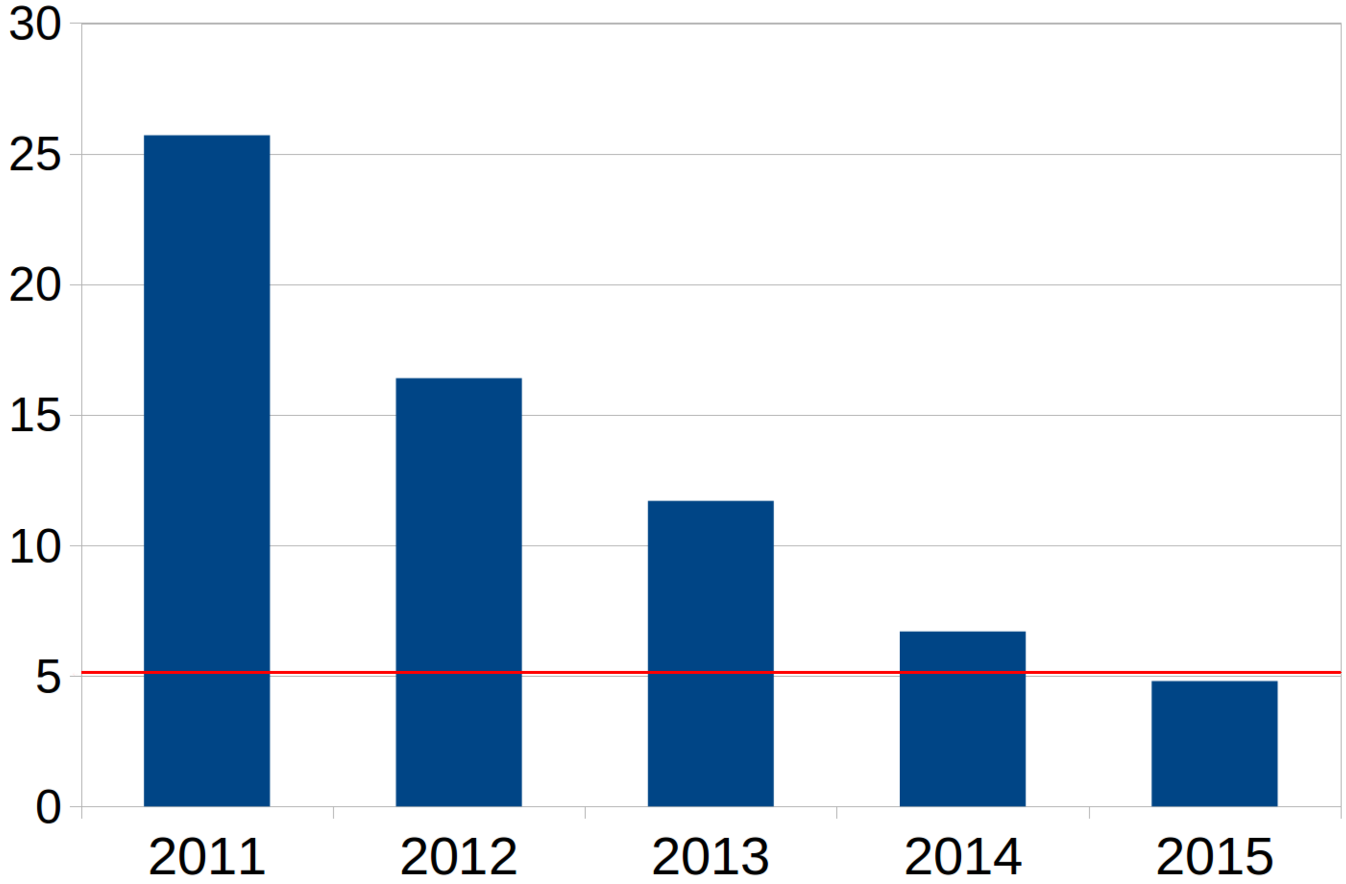
Benchmarking Beyond Branin

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

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

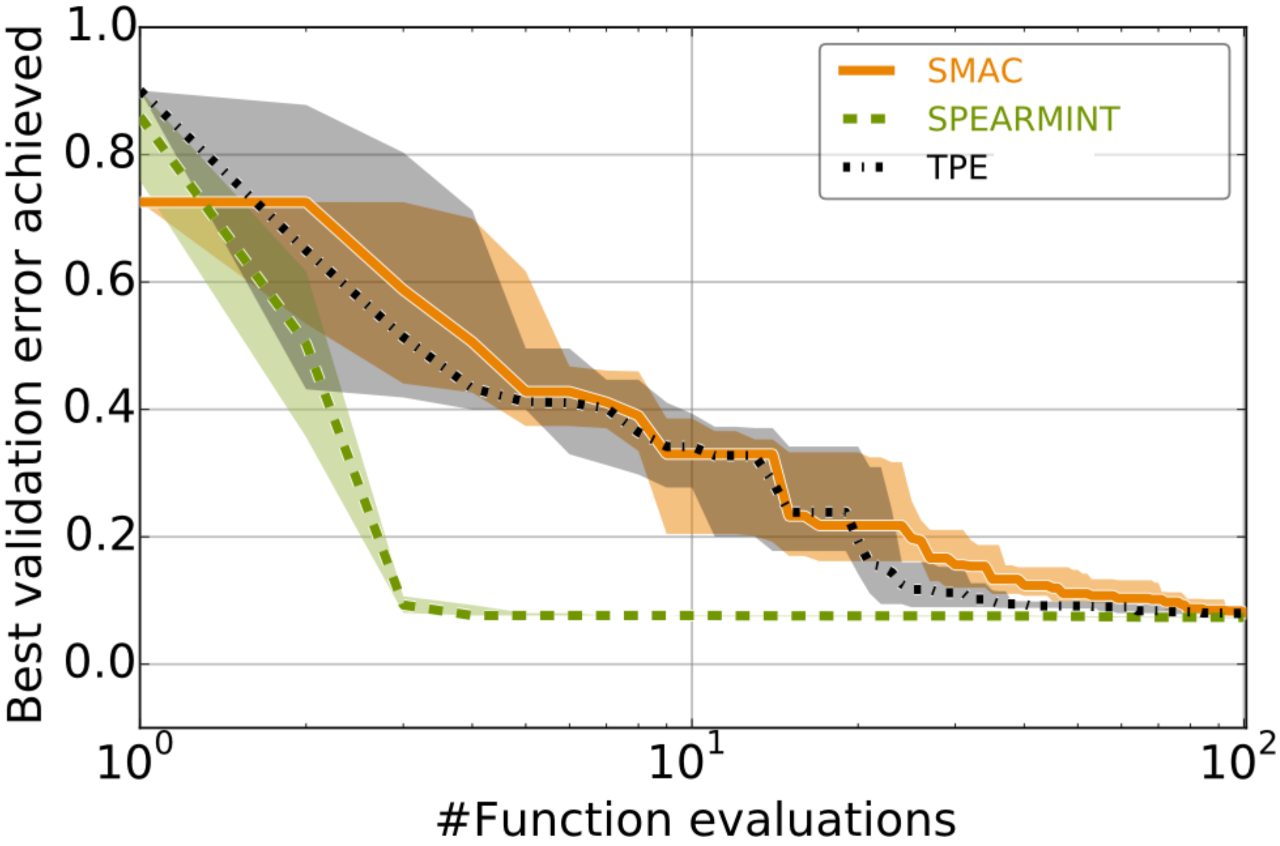
Eggensperger, Feurer, Hutter, Bergstra, Snoek , Hoos, Leyton-Brown: "**Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters**", BayesOpt'13

HPOlib: What we have learned

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

4 continuous hyperparameters

>1.5h for 100 function evaluations



Eggensperger, Feurer, Hutter, Bergstra, Snoek , Hoos, Leyton-Brown: "**Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters**", BayesOpt'13

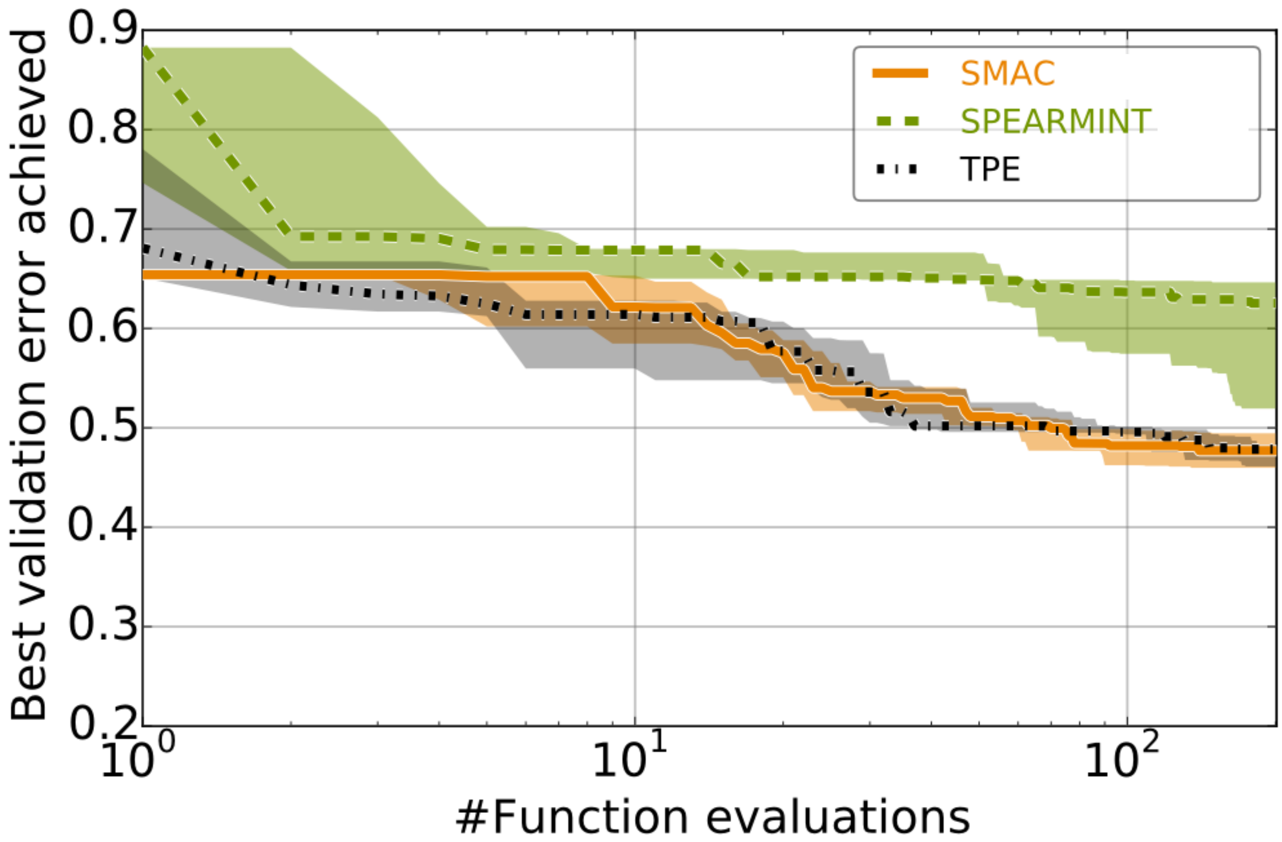
HPOlib: What we have learned

HP-DBNET on MRBI

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

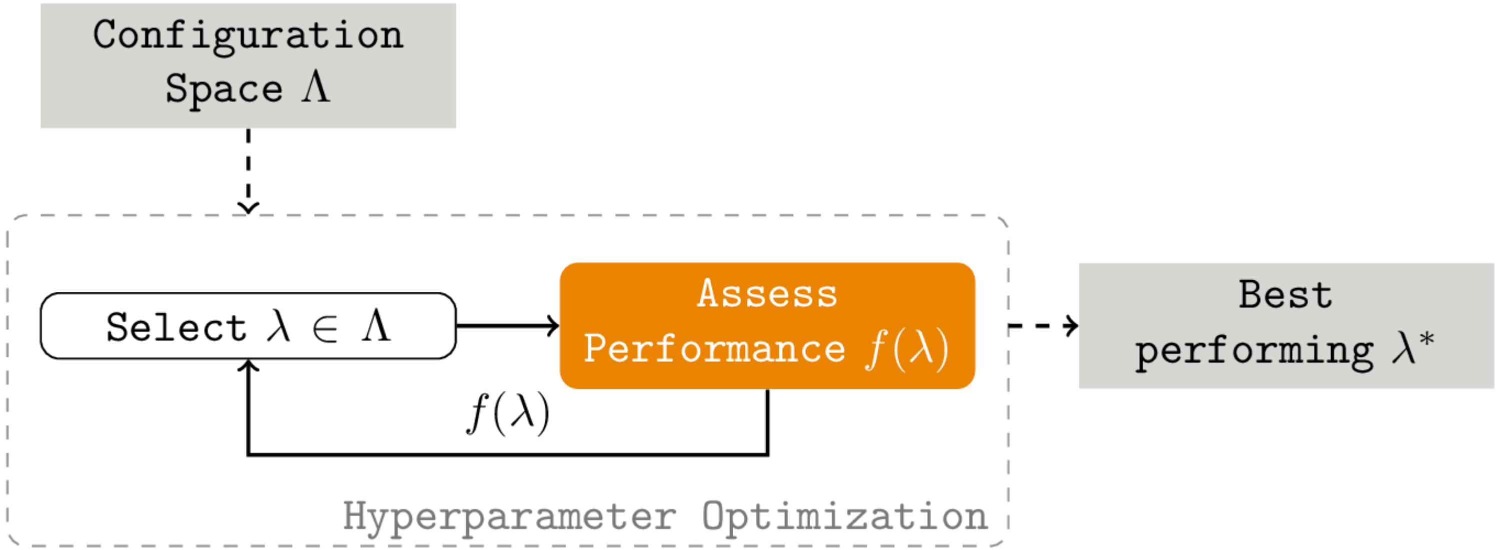
36 mixed continuous and discrete hyperparameters

>48h for 200 function evaluations on a GPU

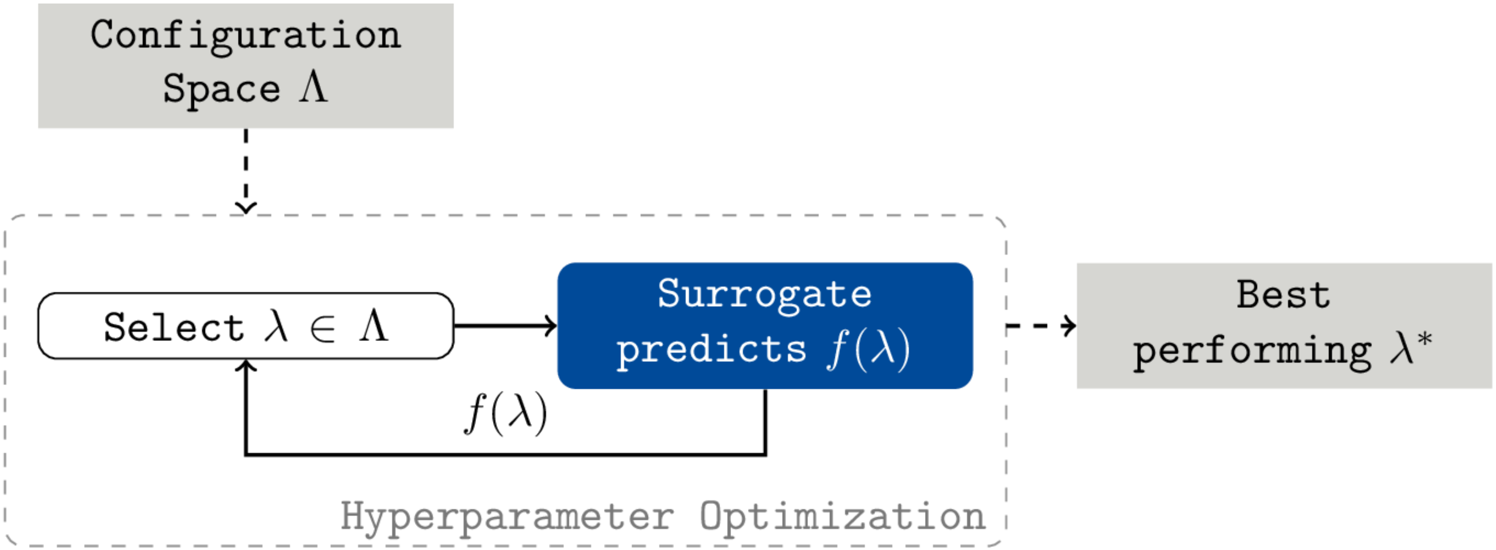


Eggensperger, Feurer, Hutter, Bergstra, Snoek , Hoos, Leyton-Brown: "**Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters**", BayesOpt'13

Speed up your experiments



Speed up your experiments



How to construct the surrogate?

1. **Collect $< \lambda, f(\lambda) >$ 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?

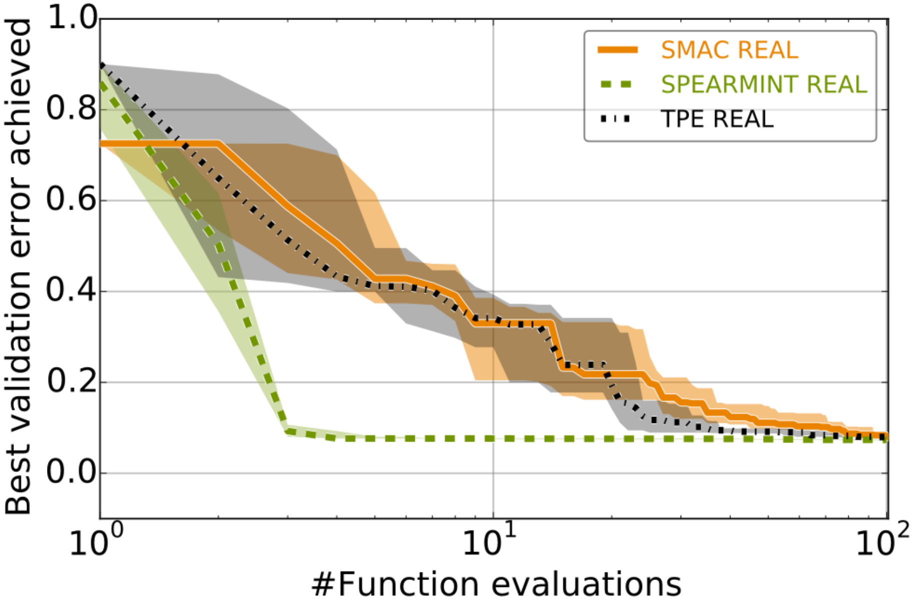
1. **Collect $< \boldsymbol{\lambda}, \boldsymbol{f}(\boldsymbol{\lambda}) >$ pairs that**
   - cover configuration space with a focus on **high-performing regions**

   → **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, …**

Surrogate Benchmarks



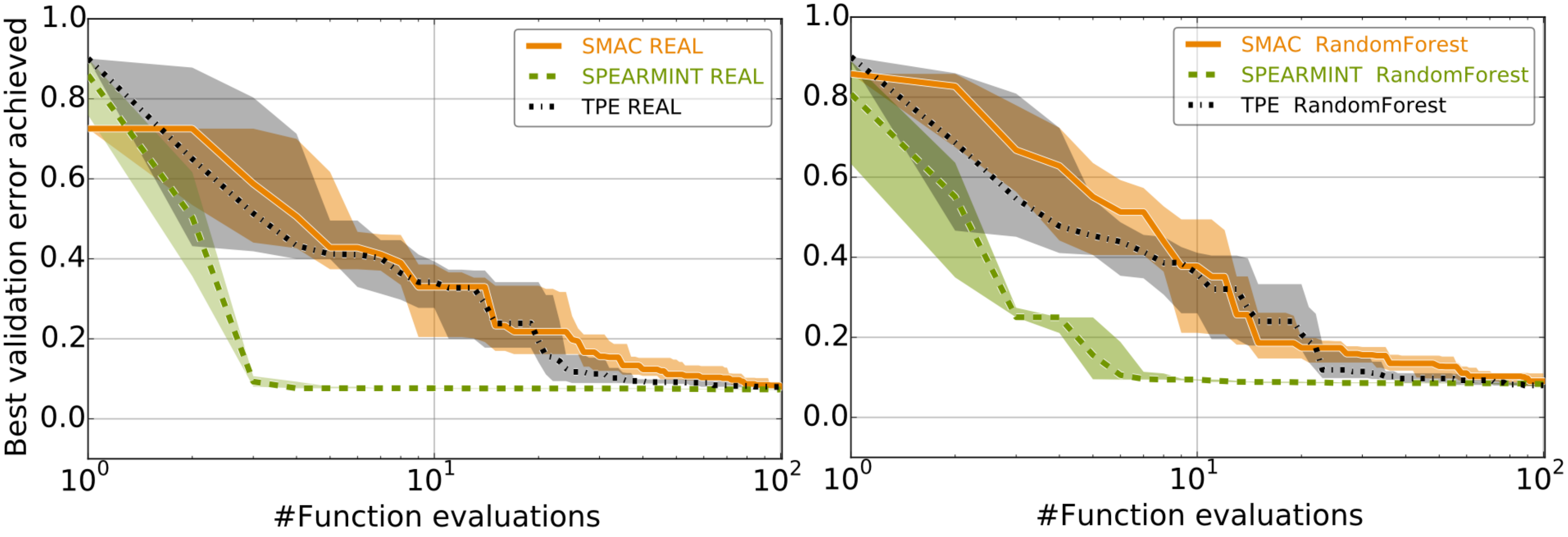
Logistic Regression on MNIST

[Snoek et al, 2012; Y. LeCun et al, 1998]

- 4 continuous hyperparameters

Eggensperger, Hutter, Hoos, Leyton-Brown: "**Efficient Benchmarking of Hyperparameter Optimizers via Surrogates**", AAAI'15

Surrogate Benchmarks



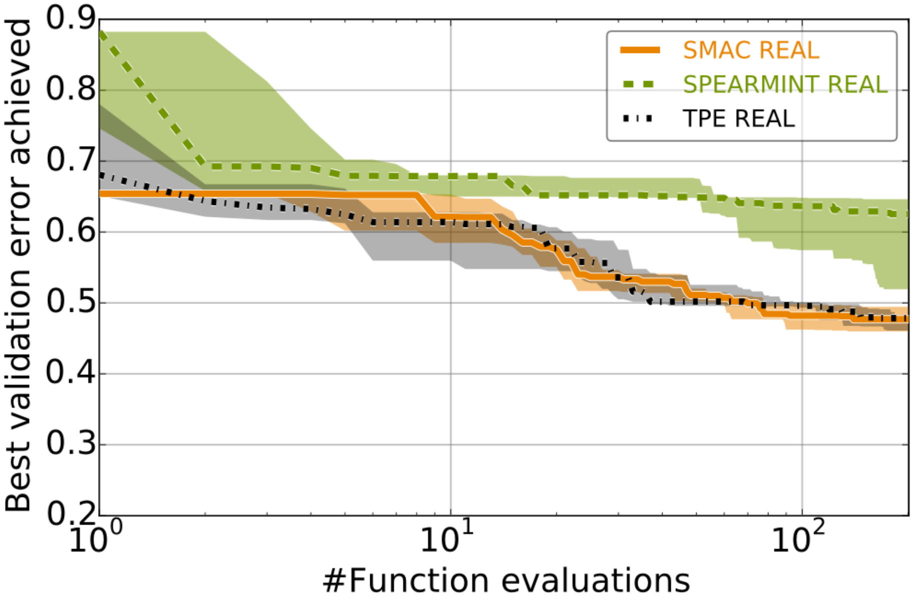
Logistic Regression on MNIST

[Snoek et al, 2012; Y. LeCun et al, 1998]

- 4 continuous hyperparameters

Eggensperger, Hutter, Hoos, Leyton-Brown: "**Efficient Benchmarking of Hyperparameter Optimizers via Surrogates**", AAAI'15

Surrogate Benchmarks



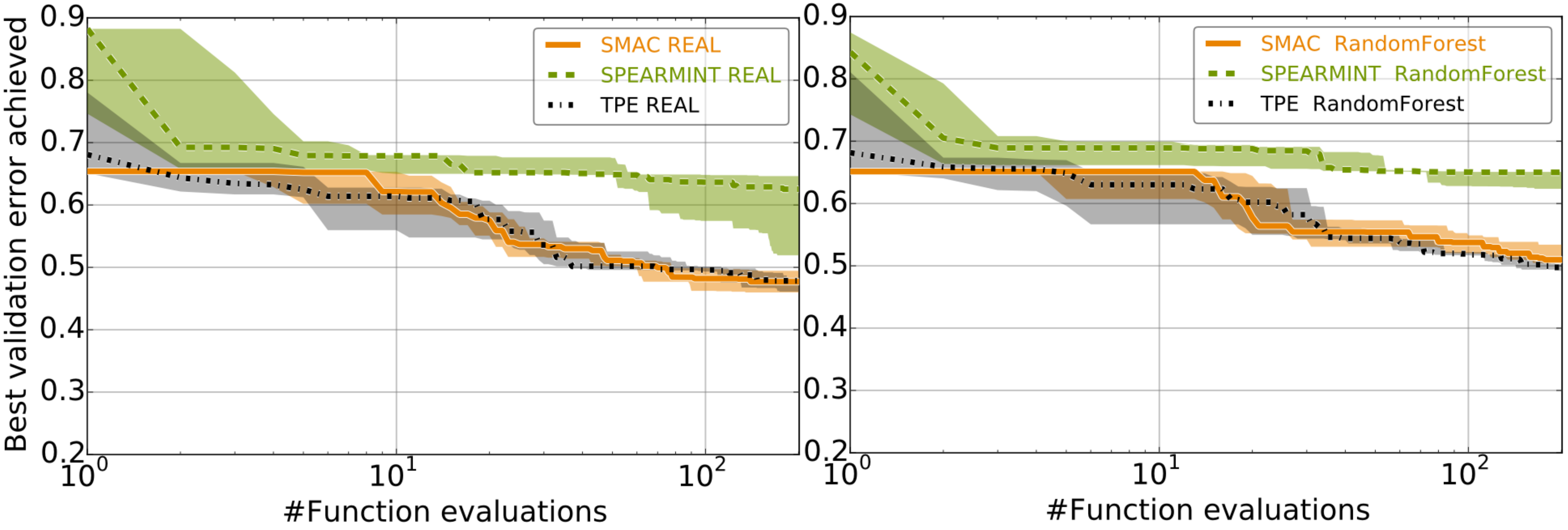
HP-DBNET on MRBI

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

- 36 mixed hyperparameter

Eggensperger, 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**

Eggensperger, 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

	Artificial Functions	Hyperparameter Optimization Problems
Realistic	x	✔
easy to set up	✔	x
cheap to run	✔	x
runnable w/o special hardware	✔	x

HPOlib: Summary

- Every optimization method has strengths and weaknesses

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

	Artificial Functions	HPOlib
Realistic	x	✔
easy to set up	✔	✔
cheap to run	✔	x
runnable w/o special hardware	✔	x

HPOlib: Summary

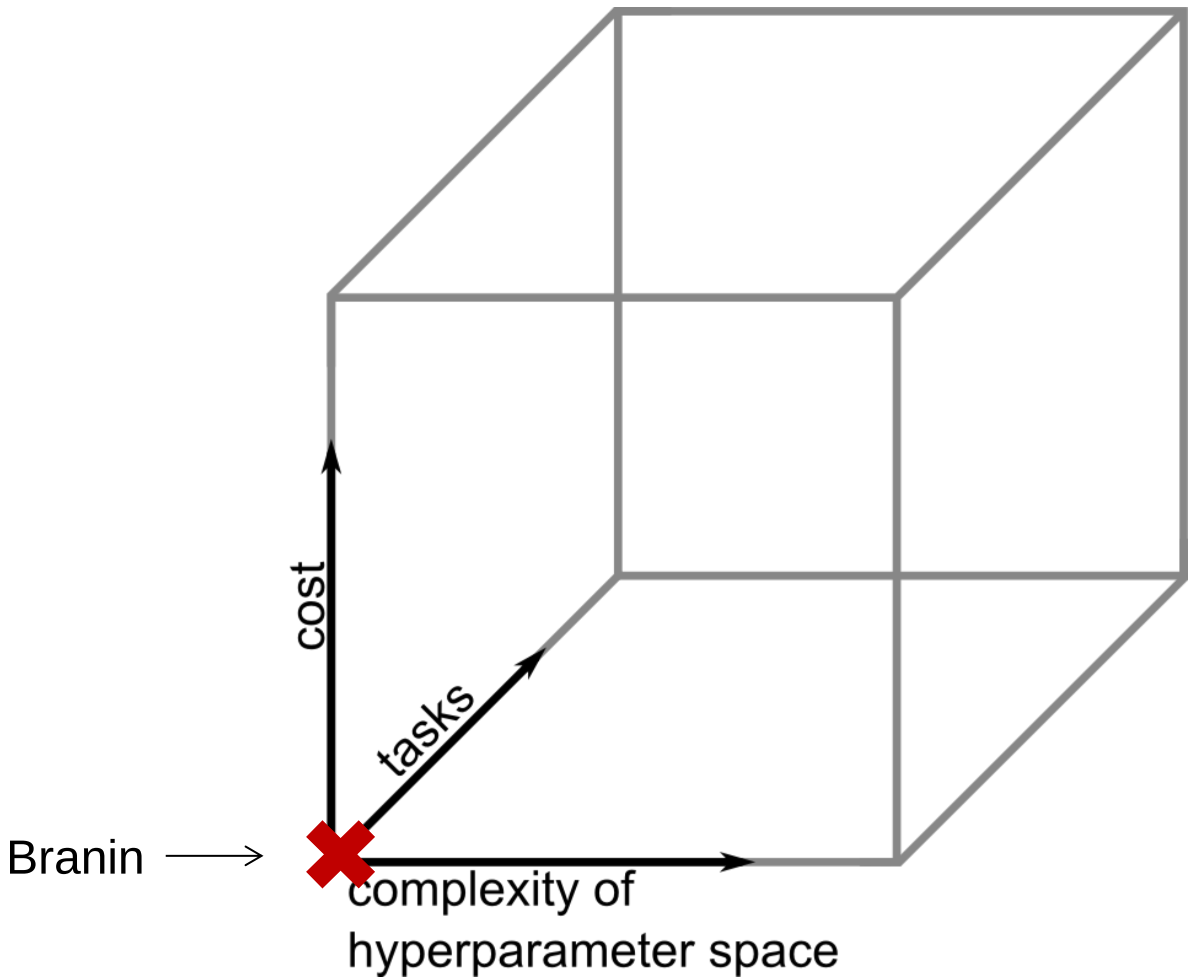
- Every optimization method has strengths and weaknesses

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

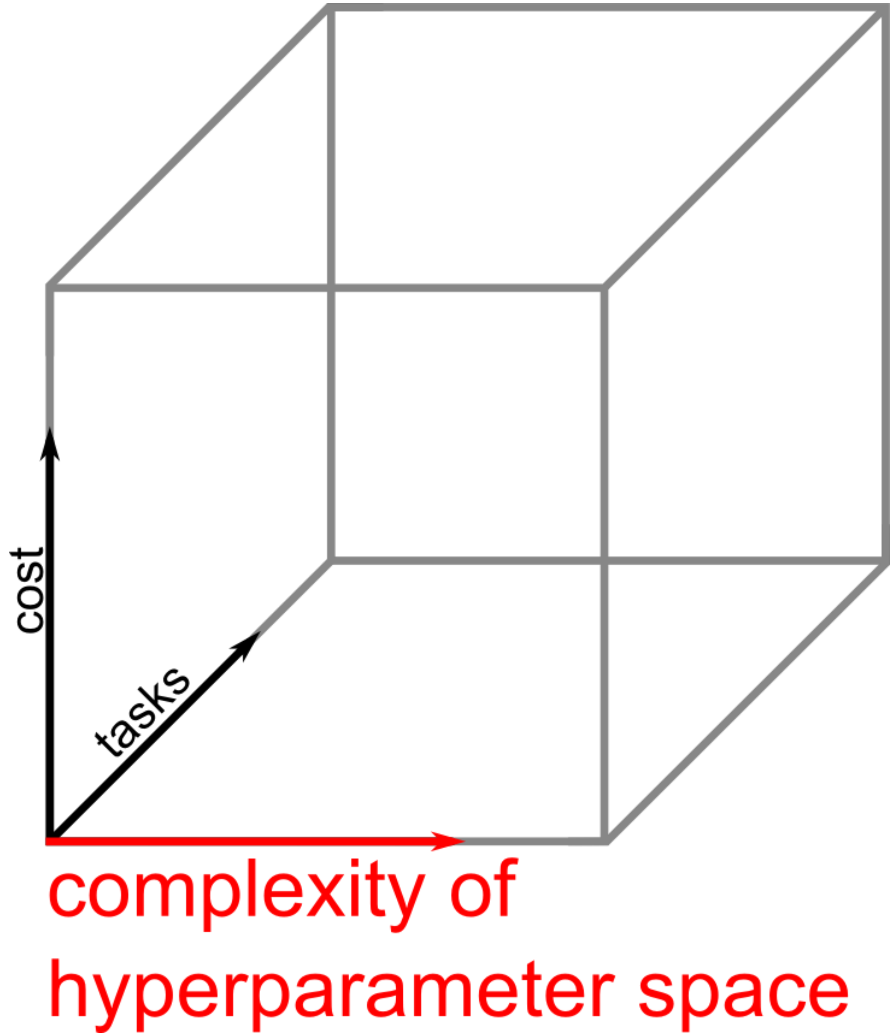
	Artificial Functions	Surrogate benchmarks
Realistic	x	✔
easy to set up	✔	✔
cheap to run	✔	✔
runnable w/o special hardware	✔	✔

UNI FREIBURG

Going Beyond



Going Beyond

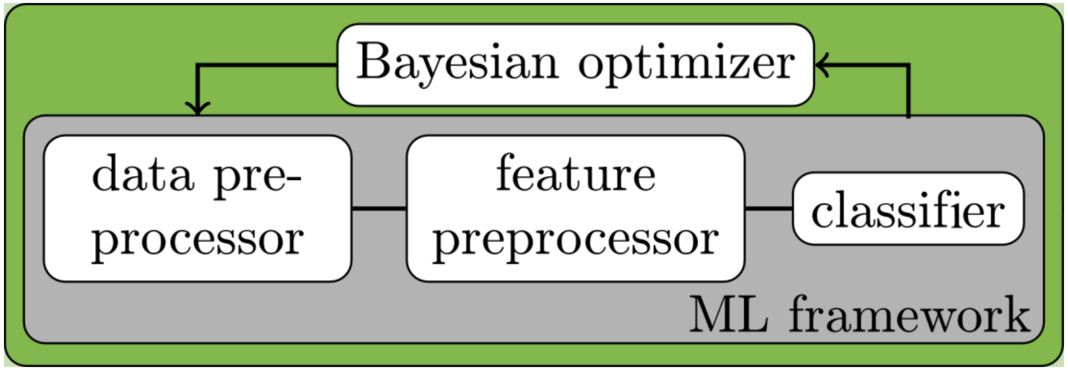


Complex Hyperparameter Spaces

- 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

one-hot encoding	2
imputation	1
balancing	1
rescaling	1

Auto-sklearn's Configuration Space

name	#λ
extreml. rand. trees prepr.	5
fast ICA	4
feature agglomeration	4
kernel PCA	5
rand. kitchen sinks	2
linear SVM prepr.	3
no preprocessing	-
nystroem sampler	5
PCA	2
polynomial	3
random trees embed.	4
select percentile	2
select rates	3

one-hot encoding	2
imputation	1
balancing	1
rescaling	1

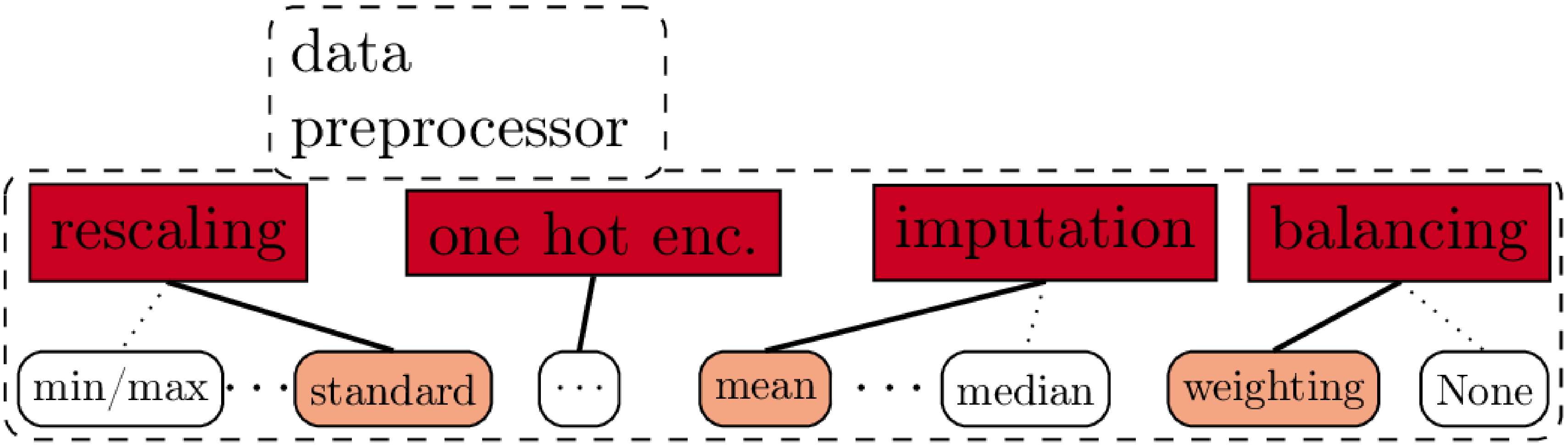
Auto-sklearn's Configuration Space

name	#λ
extreml. rand. trees prepr.	5
fast ICA	4
feature agglomeration	4
kernel PCA	5
rand. kitchen sinks	2
linear SVM prepr.	3
no preprocessing	-
nystroem sampler	5
PCA	2
polynomial	3
random trees embed.	4
select percentile	2
select rates	3

one-hot encoding	2
imputation	1
balancing	1
rescaling	1

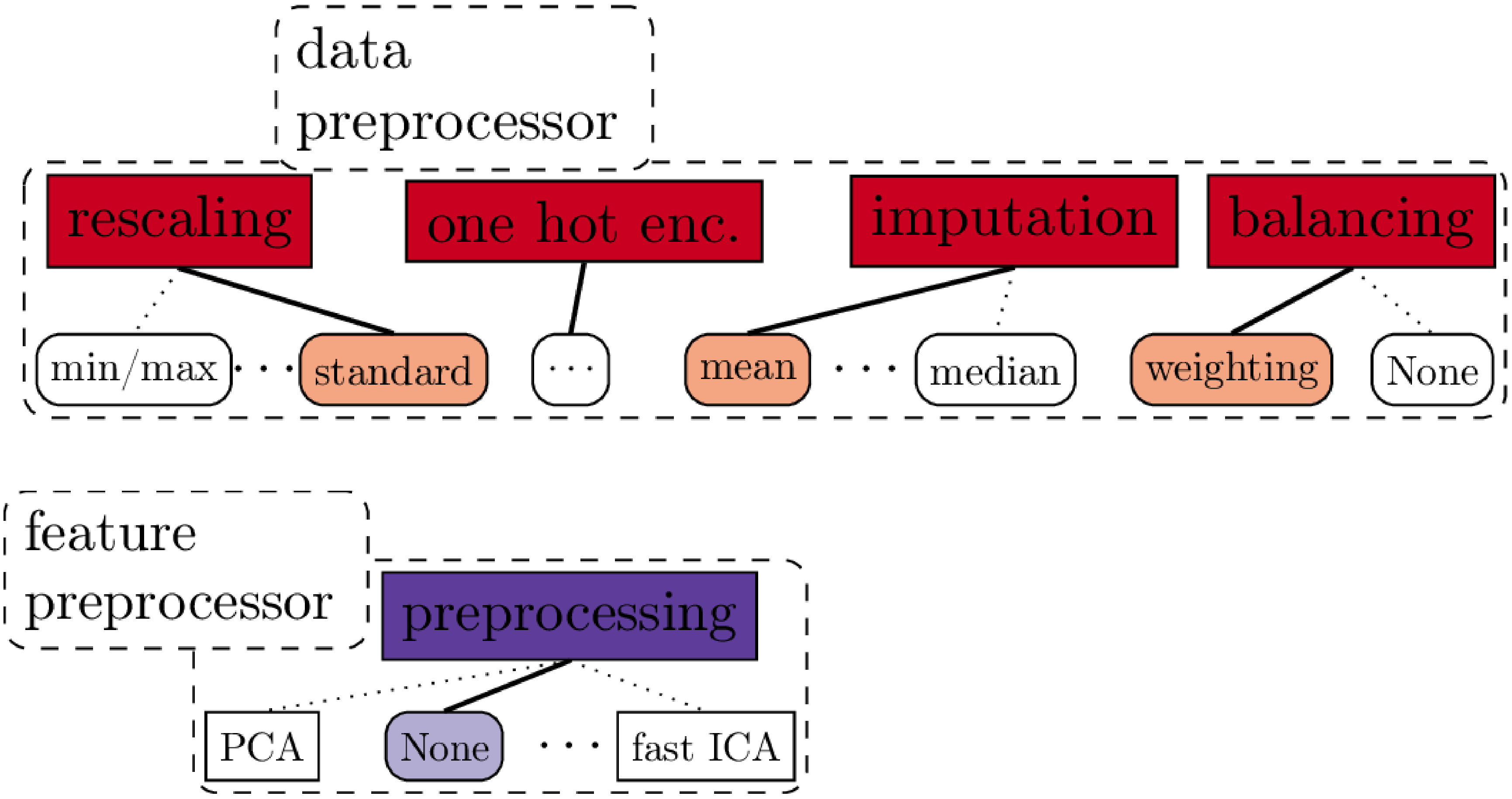
name	#λ
AdaBoost (AB)	4
Bernoulli naïve Bayes	2
decision tree (DT)	4
extreml. rand. trees	5
Gaussian naïve Bayes	-
gradient boosting (GB)	6
kNN	3
LDA	4
linear SVM	4
kernel SVM	7
multinomial naïve Bayes	2
passive aggressive	3
QDA	2
random forest (RF)	5
Linear Class. (SGD)	10

Auto-sklearn's Configuration Space



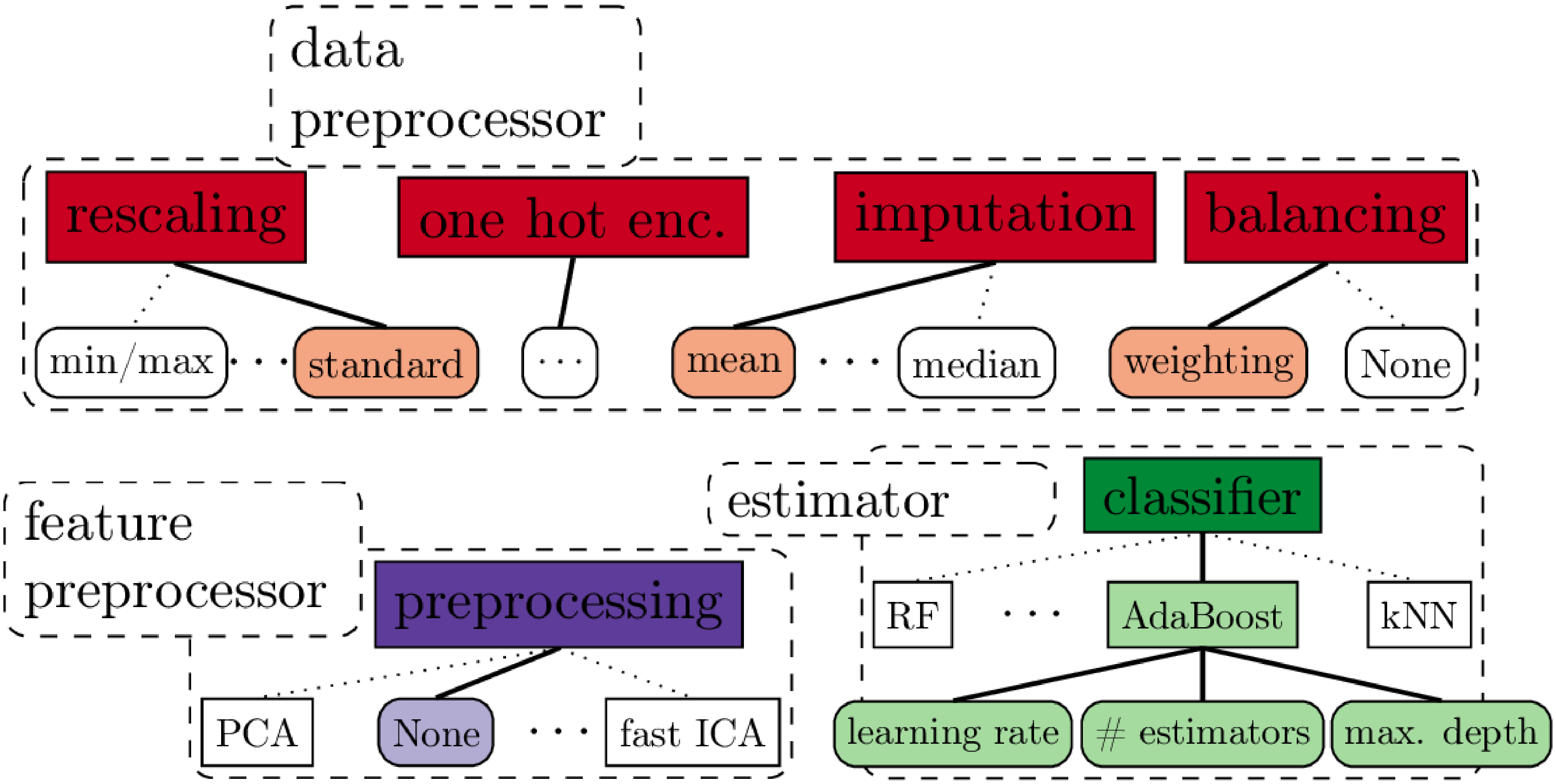
M. Feurer and A. Klein and K. Eggensperger and J. Springenberg and M. Blum and F. Hutter: "**Efficient and Robust Automated Machine Learning**", NIPS'15

Auto-sklearn's Configuration Space



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

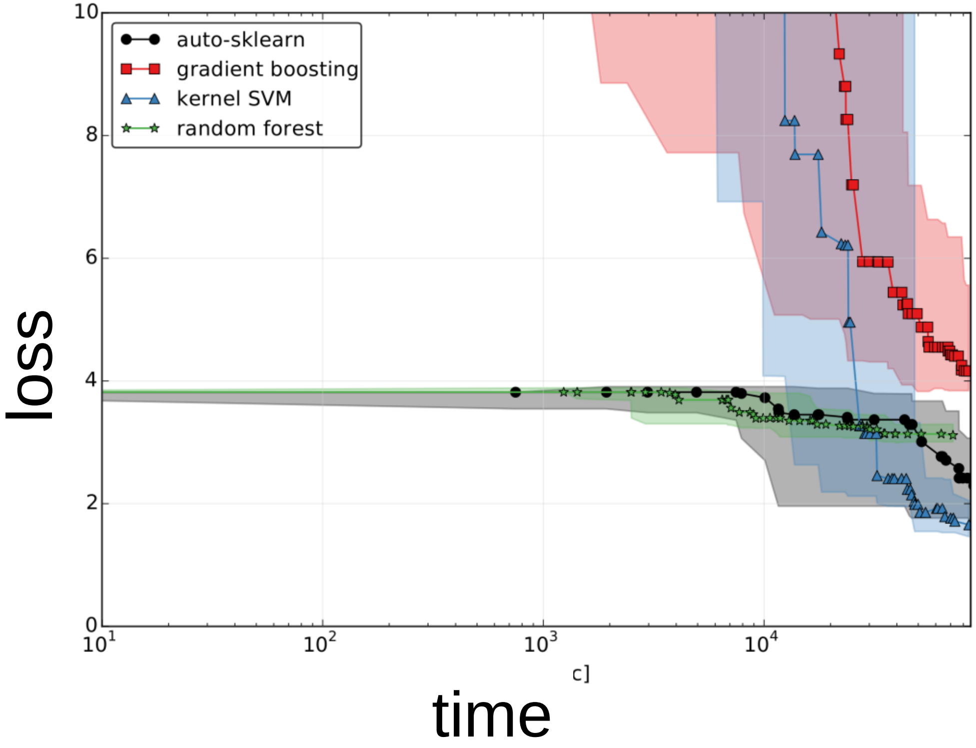
Auto-sklearn's Configuration Space



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

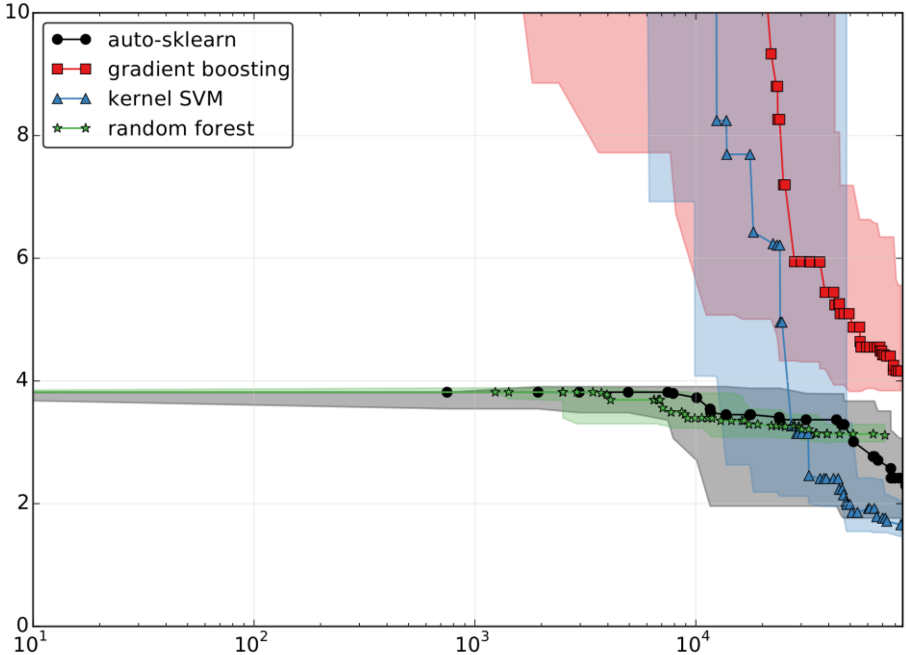
Bayesian Optimization can handle that space

MNIST

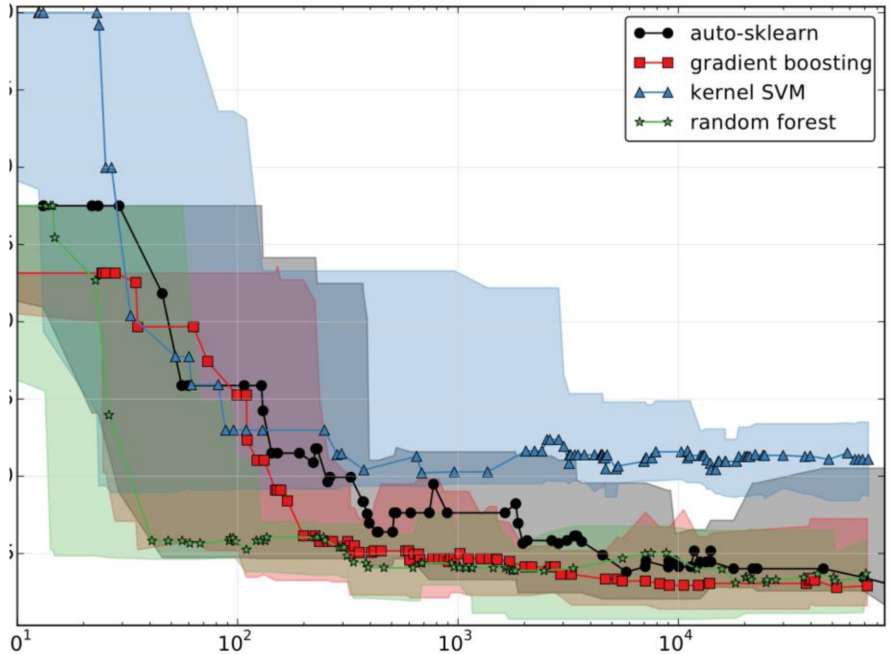


Bayesian Optimization can handle that space

MNIST



PROMISE PC4

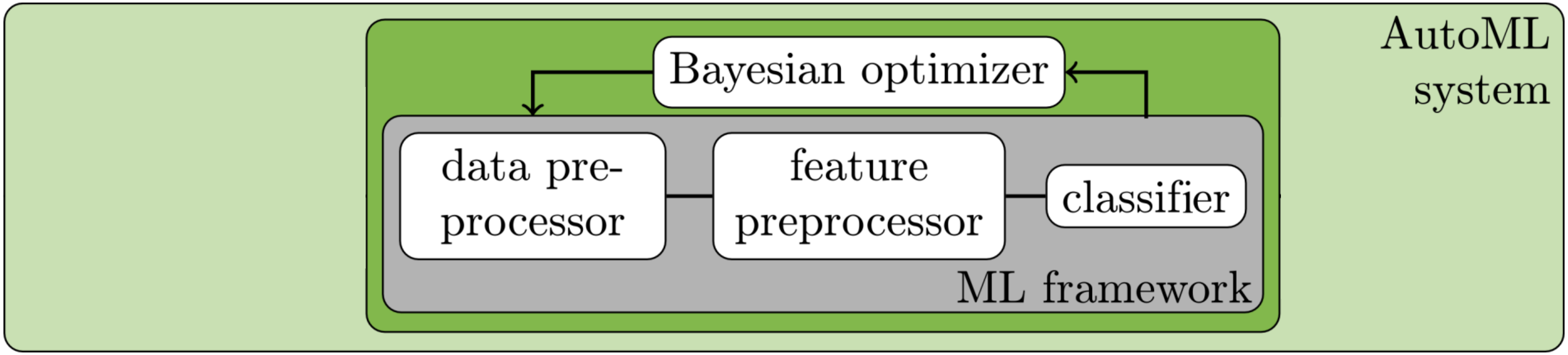


loss

time

time

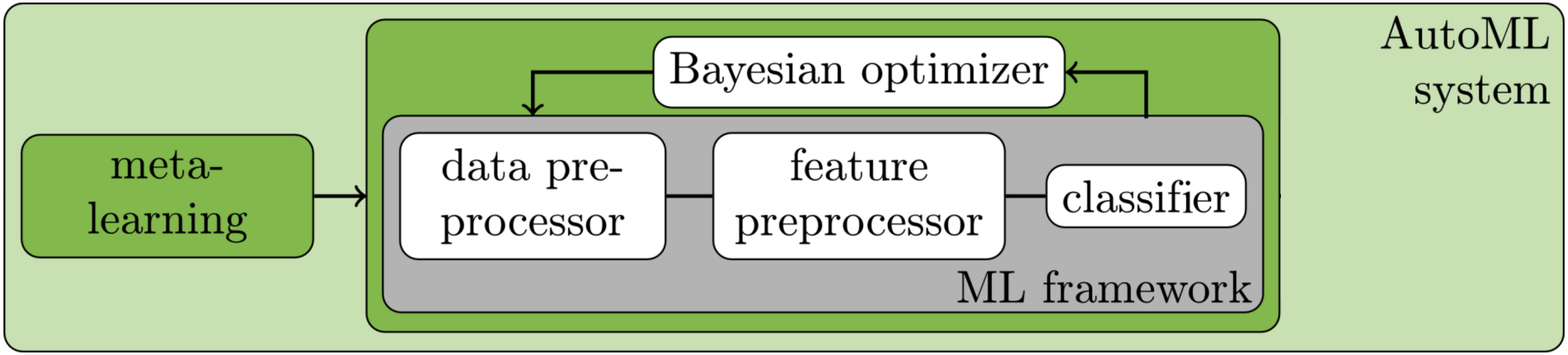
Auto-sklearn Workflow



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

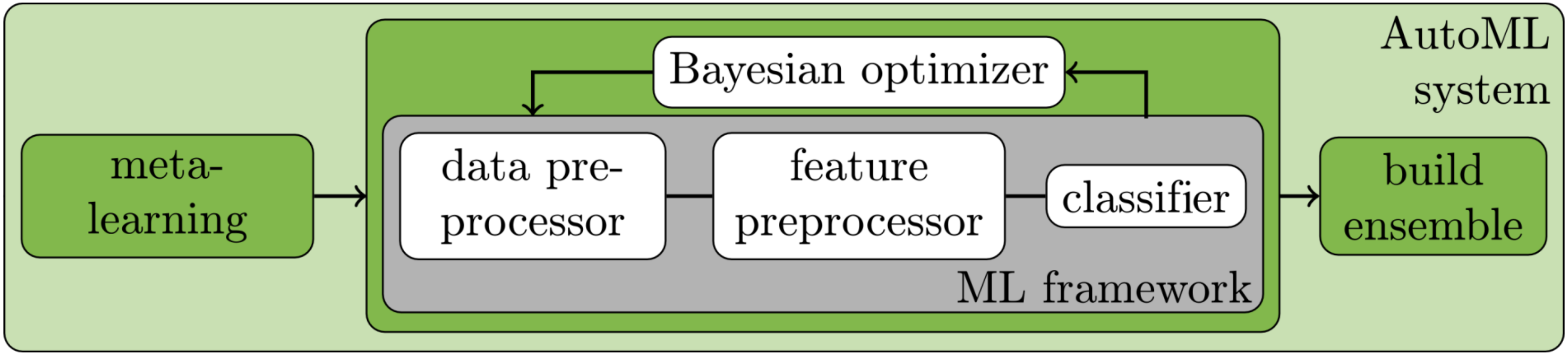
Auto-sklearn Workflow



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

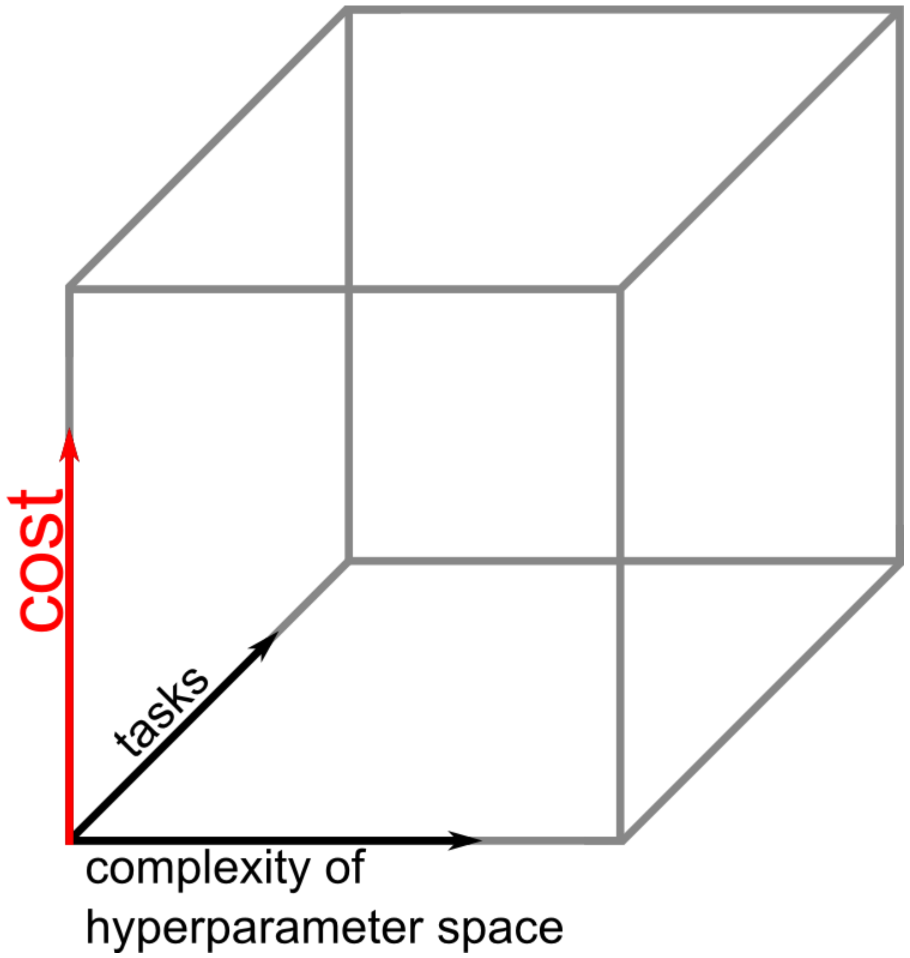
Auto-sklearn Workflow



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



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

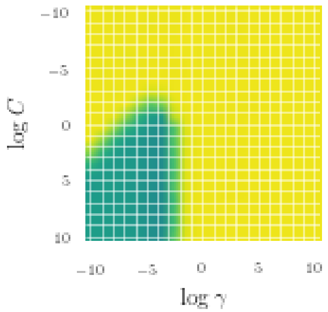
Recent approaches:

- 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

Fork me on GitHub

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

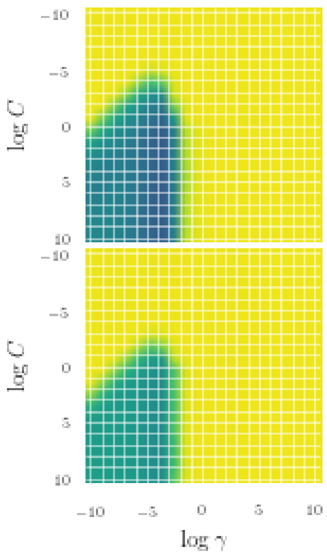


https://github.com/automl/RoBO

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

Dataset size



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

https://github.com/automl/RoBO

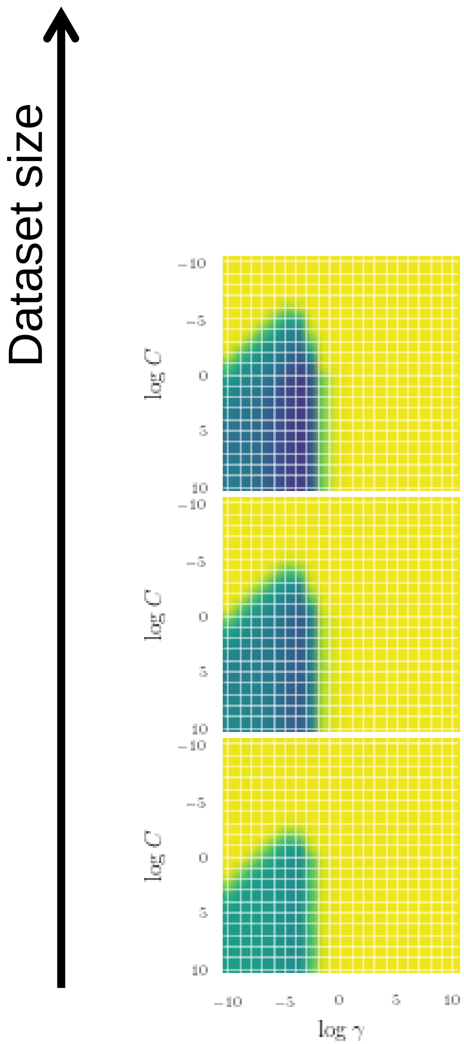
A. Klein and S. Falkner and S. Bartels and P. Hennig and F. Hutter: **"Fast Bayesian optimization of Machine Learning Hyperparameters on Large Datasets"**, ArXiv'16

Fork me on GitHub

ML4AAD

Fork me on GitHub

Example: Fabolas

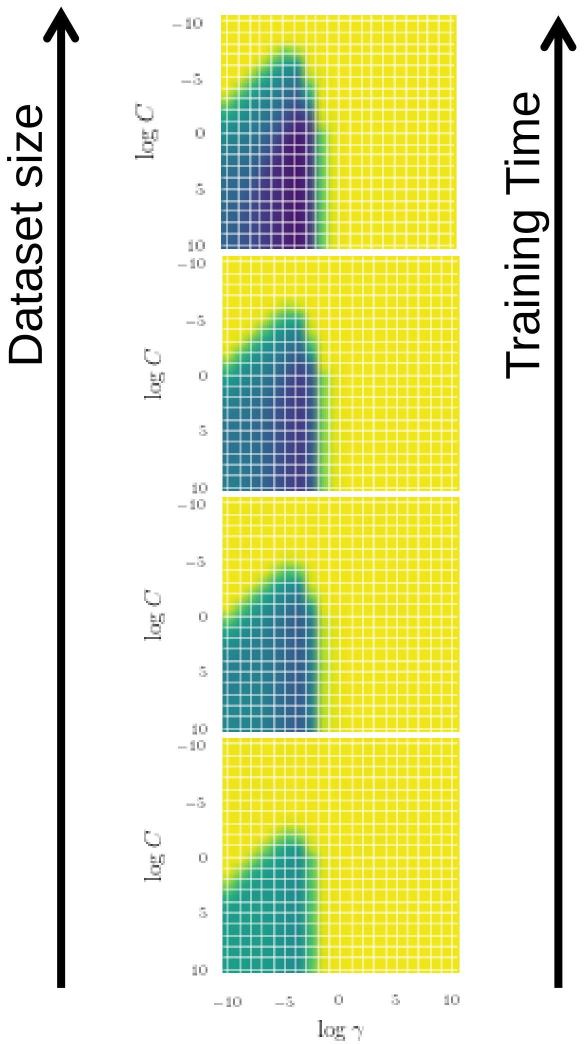


FAst **B**ayesian **O**ptimization on **LA**rge Data **S**ets

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



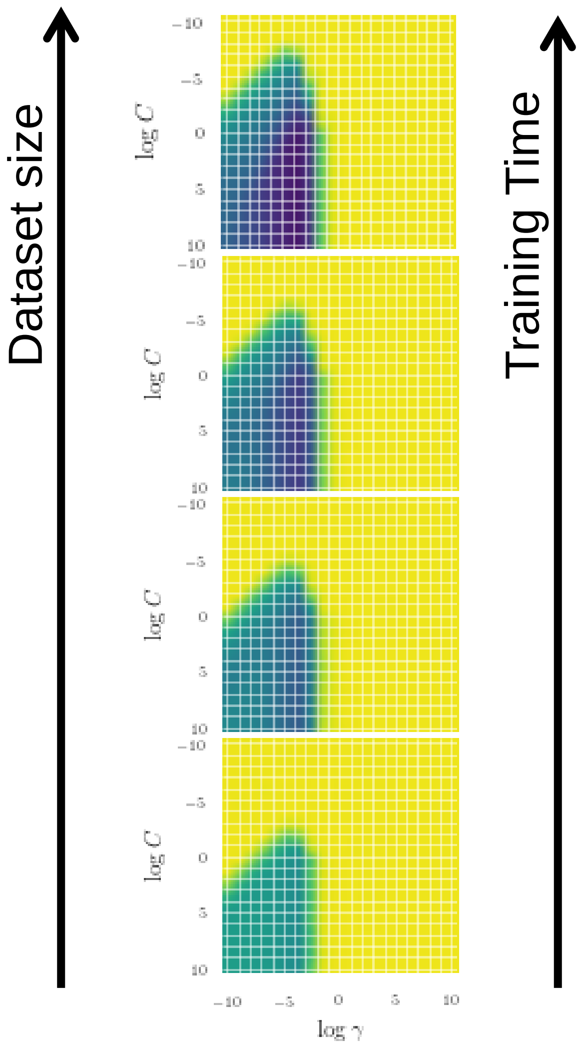
FAst **B**ayesian **O**ptimization on **LA**rge Data **S**ets

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

Fork me on GitHub



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

Small data subsets suffice to estimate performance of a configuration

→ Model data set size as an additional degree of freedom

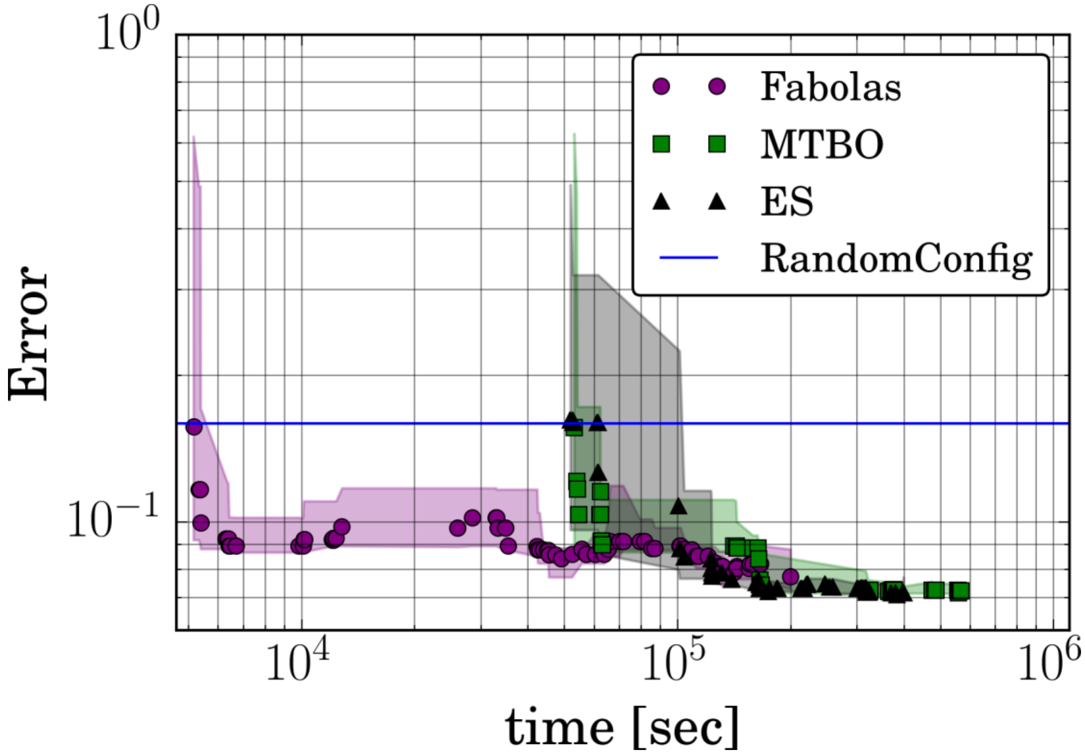
https://github.com/automl/RoBO

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

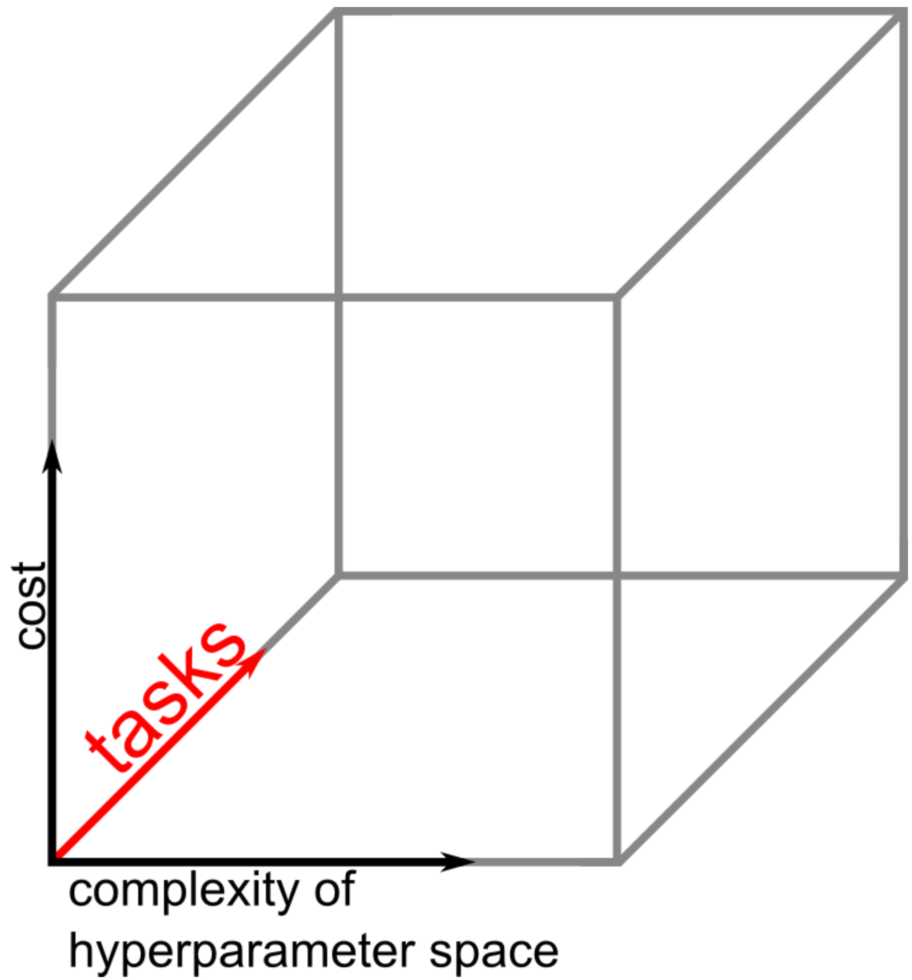
ResNet on CIFAR-10
[He et al, 2015]

- 4 continuous hyperparameters



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

Going Beyond



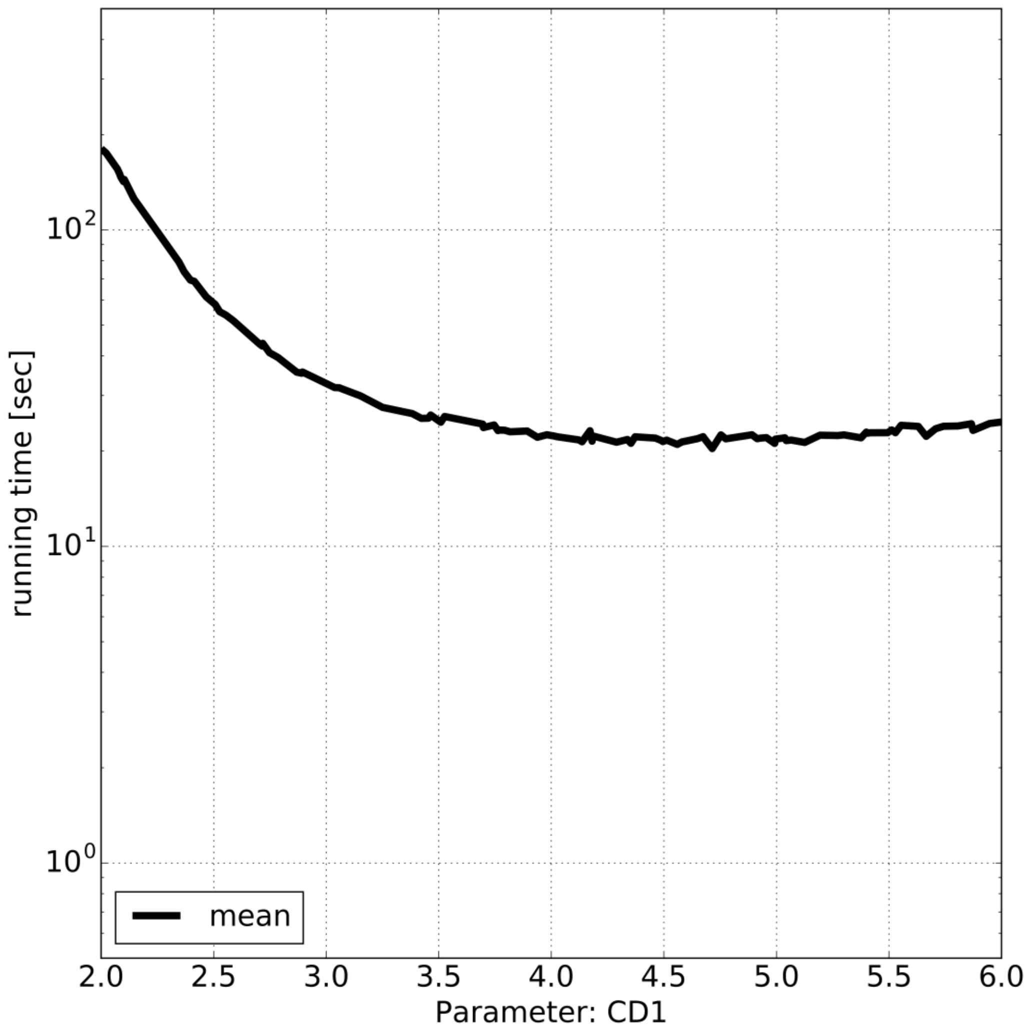
Optimizing across tasks

- Challenges:
  - Tune hyperparameters across a set of tasks

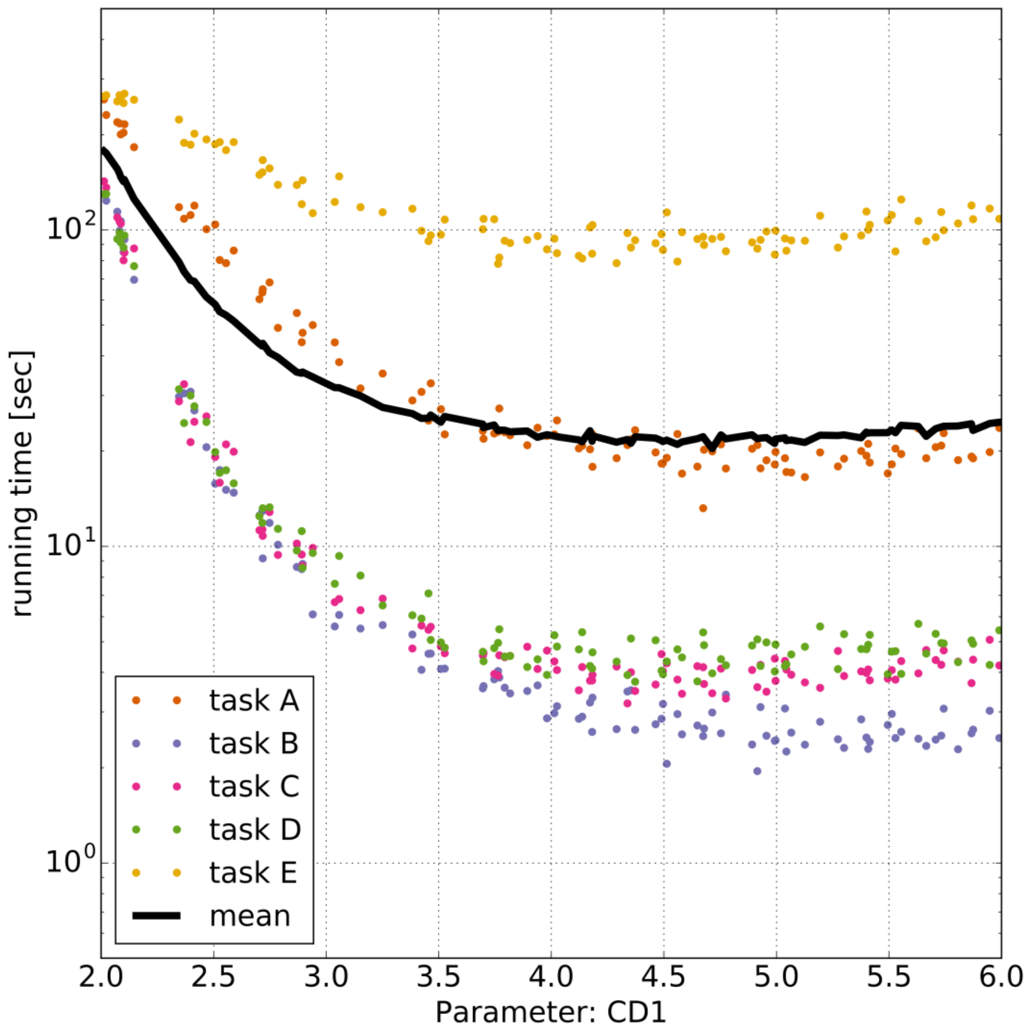
$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} \, \mathbb{E}_{t \in T} f_t(\lambda)$$

- Examples:
  - Hyperparameter tuning across crossvalidation folds
  - General algorithm configuration

Example: ProbSAT on 7Sat90

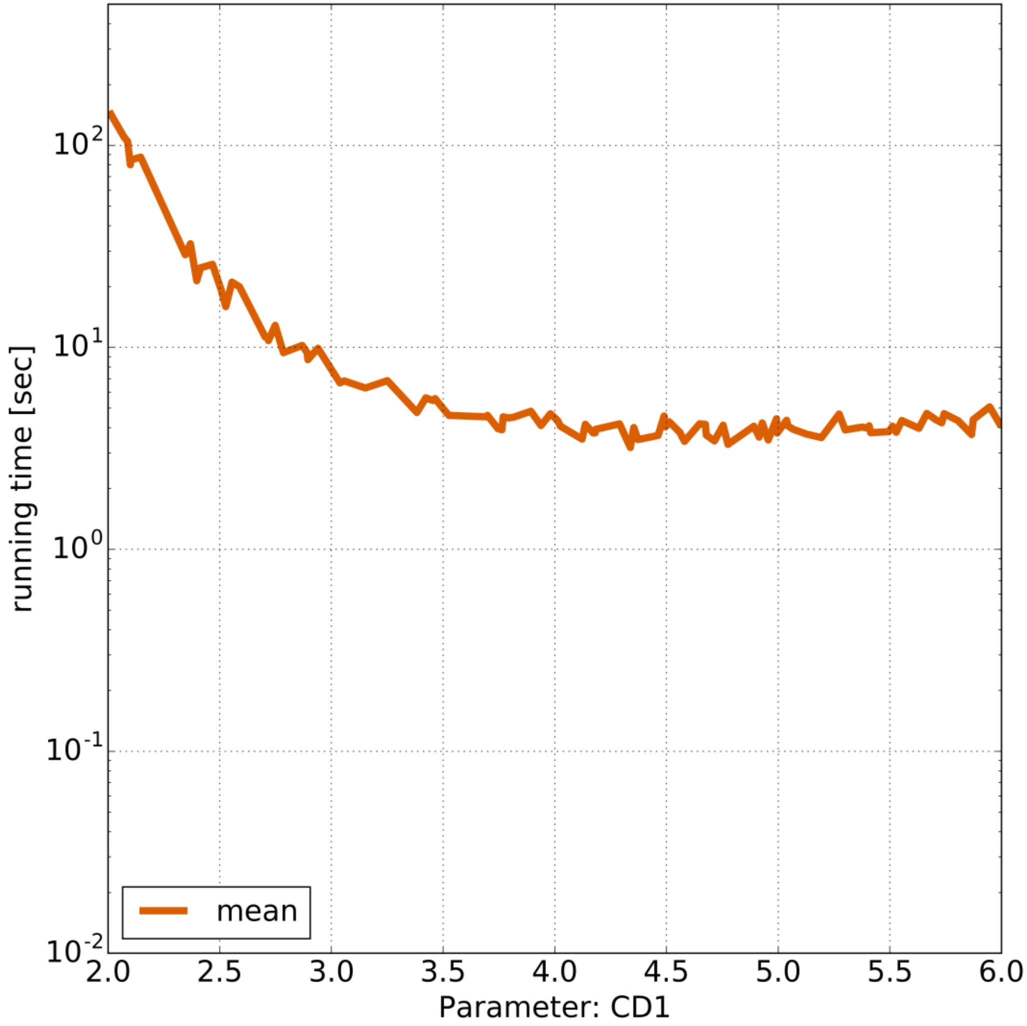


Example: ProbSAT on 7Sat90

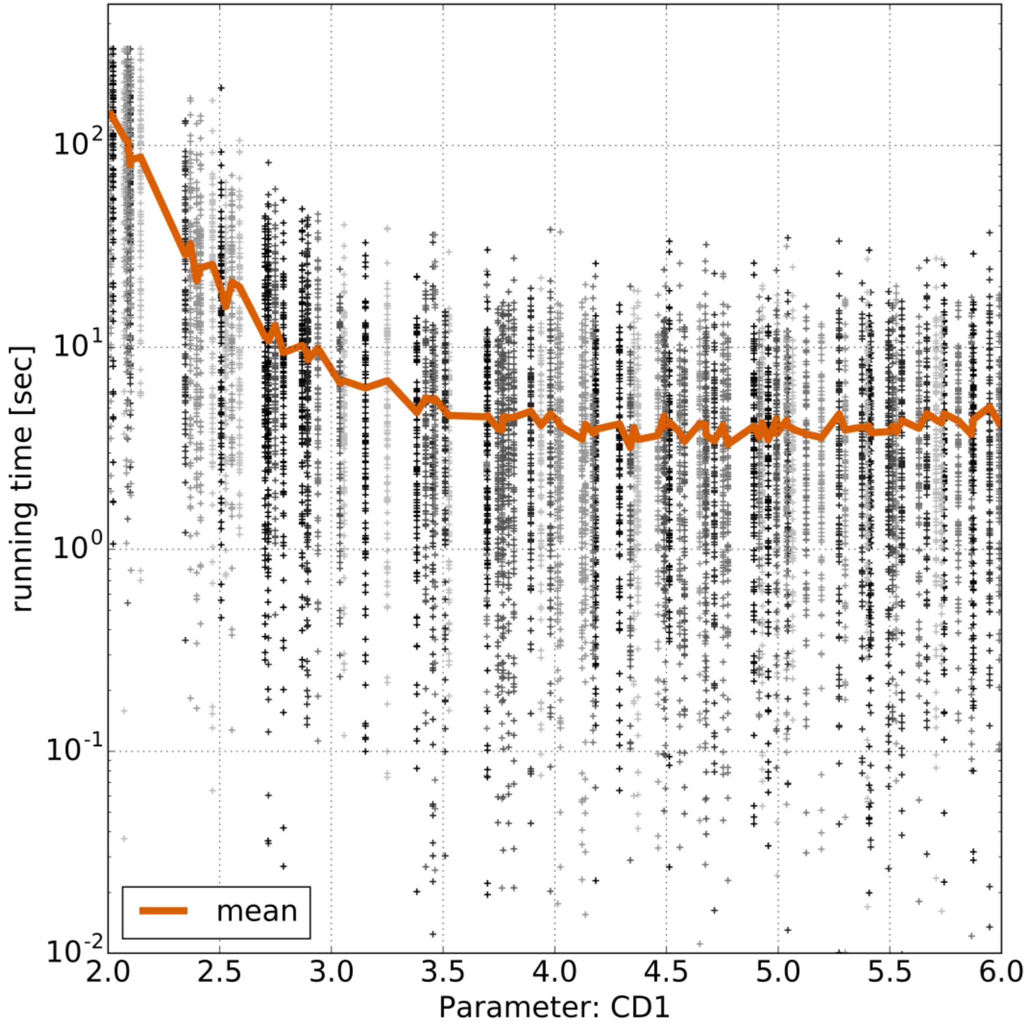


→ **Objective value varies across tasks**

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

Fork me on ~~GitHub~~ bitbucket

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

https://bitbucket.org/mlindauer/aclib2

Can we build surrogate models for these benchmark problems, too?

How to construct the surrogate?

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

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

How to construct the surrogate?

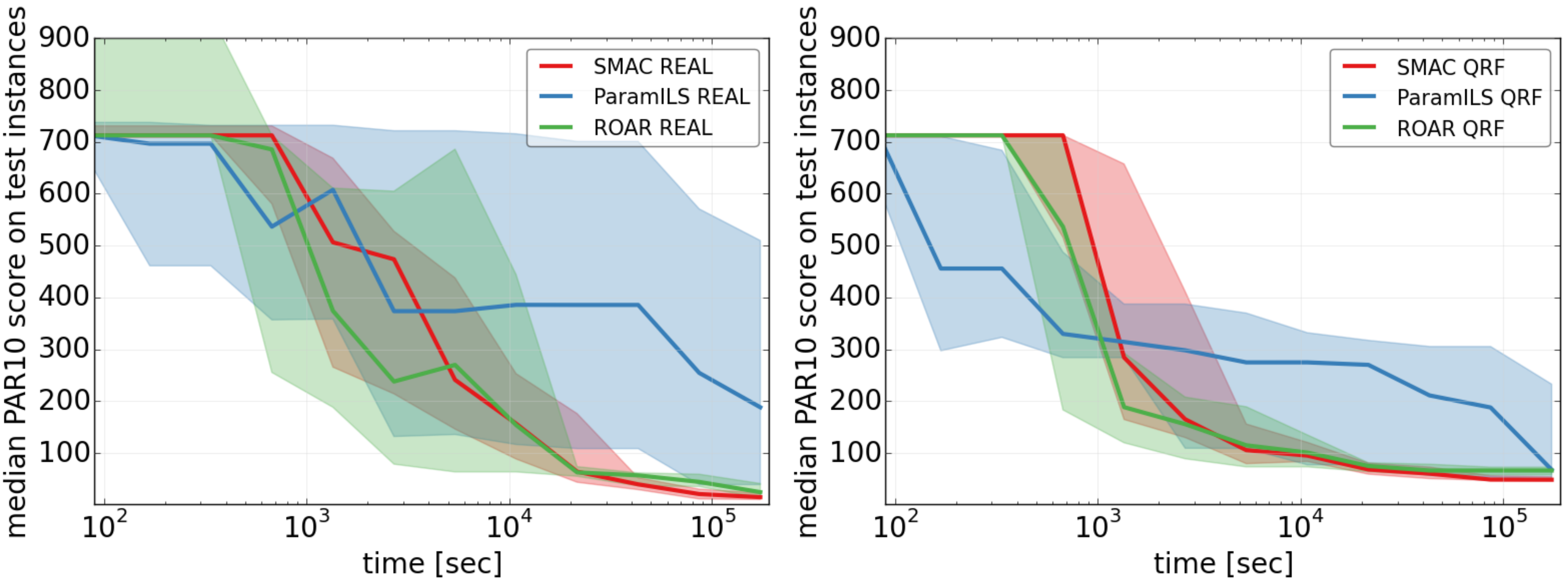
1. **Collect $< \boldsymbol{\lambda}, \boldsymbol{t}, \boldsymbol{f}(\boldsymbol{\lambda}, \boldsymbol{t}) >$ tuples that**
   - cover configuration space with a focus on **high-performing regions**

   → **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

Algorithm	#hyper-parameter	Dataset
Artificial Functions	1-X	-
Auto-sklearn	>100	OpenML
Logistic Regression	4	MNIST
SVM	2	MNIST
ResNet	4	CIFAR-10
ConvNet	5	CIFAR-10
Fully Connected Net	10	MNIST
…		

https://github.com/automl/HPOlib2

Fork me on GitHub

HPOlib2: easy-to-use

Fork me on GitHub

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

UNI FREIBURG

HPOlib2: easy-to-use

Fork me on GitHub

```
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()
```

UNI
FREIBURG

HPOlib2: easy-to-use

Fork me on GitHub

```
from hpolib.benchmarks.ml import svm_benchmark

# Download datasets
b = svm_benchmark.SvmOnMnist()

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

HPOlib2: easy-to-use

Fork me on GitHub

```
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}
```

HPOlib2: easy-to-use

```
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}
```

Fork me on GitHub

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