Hyperparameter Optimization and AutoML

Bernd Bischl

SLDS - LMU Munich

Marius Lindauer

TNT - Leibniz University Hannover

www.slds.stat.uni-muenchen.de

www.tnt.uni-hannover.de

ML BASICS - RISK MINIMIZATION

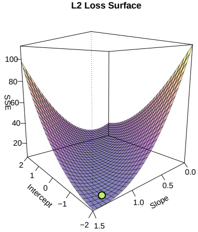
Given

- Data $\mathcal{D} = \left(\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right) \in (\mathcal{X} \times \mathcal{Y})^n$

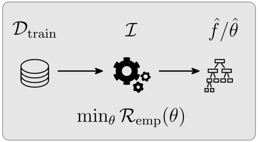
Choice

- Hypo space $\mathcal{H}$ with candidate model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^g$
- Loss $L : \mathcal{Y} \times \mathbb{R}^g \to \mathbb{R}_0^+$; defines empirical risk

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)$$



- Usually some regularization term to constrain overfitting
- Some optimizer like GD



Result

- Learner is defined: $\mathcal{I} : \mathcal{D} \to \boldsymbol{\Theta}$; finds best params via:

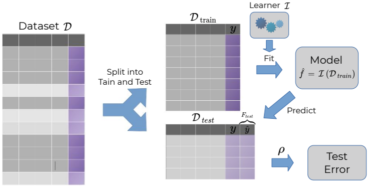
$$\hat{\boldsymbol{\theta}} \in \underset{\theta \in \boldsymbol{\Theta}}{\arg\min}\, \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$$

Conclusion: ML is neither magic, nor general AI, but parametrized curve fitting – which can be a very powerful tool

ML BASICS - GENERALIZATION ERROR

Given

- Train-test-split $\mathcal{D}_{\text{train}} \,\dot\cup\, \mathcal{D}_{\text{test}} = \mathcal{D}$
- Fitted model $\hat{f}$ from $\mathcal{D}_{\text{train}}$

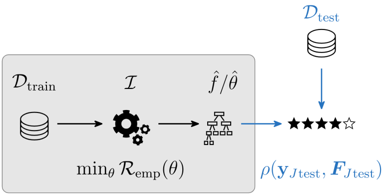


Choice

- Performance metric $\rho(\mathbf{y}_{J_{\text{test}}}, F_{J_{\text{test}}})$

  $F_{J_{\text{test}}}$ is pred-matrix w.r.t. $\mathcal{D}_{\text{test}}$ and $\mathbf{y}_{J_{\text{test}}}$ is true label vector
- Often:

$$\rho_L(\mathbf{y}_{J_{\text{test}}}, F_{J_{\text{test}}}) = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} L\left(\mathbf{y}^{(i)}_{J_{\text{test}}}, F^{(i)}_{J_{\text{test}}}\right)$$



Result

- Assessment of how well $\hat{f}$ generalizes:

$$\widehat{GE} = \rho(\mathbf{y}_{J_{\text{test}}}, F_{J_{\text{test}}})$$

Single train-test-split results in pessimistic bias and high variance of estimator $\widehat{GE}$ (both sets smaller than intended); **Resampling** (CV, subsampling, ...) repeats this process and solves this dilemma

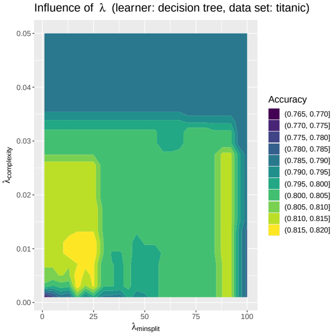
TUNING - HYPERPARAMETER OPTIMIZATION

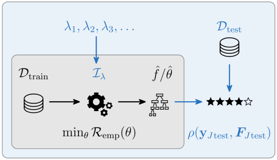
- Hyperparameter configuration $\boldsymbol{\lambda}$ configures $\mathcal{I}_{\boldsymbol{\lambda}}$ and strongly influences model quality
- Examples: Regularization constants, optimizer settings, model component types, . . .
- **Tuning** / **HPO:** Find best HPC with optimal $\widehat{GE}$

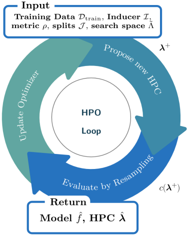
$$\hat{\boldsymbol{\lambda}} \in \operatorname*{arg\,min}_{\boldsymbol{\lambda} \in \tilde{\boldsymbol{\Lambda}}} c(\boldsymbol{\lambda}) \quad \text{with} \quad c(\boldsymbol{\lambda}) := \widehat{GE}(\mathcal{I}, \mathcal{J}, \rho, \boldsymbol{\lambda})$$

$\mathcal{J}$ is train-test splits, $\tilde{\boldsymbol{\Lambda}}$ is search space

- **Expensive, noisy, black box problem**

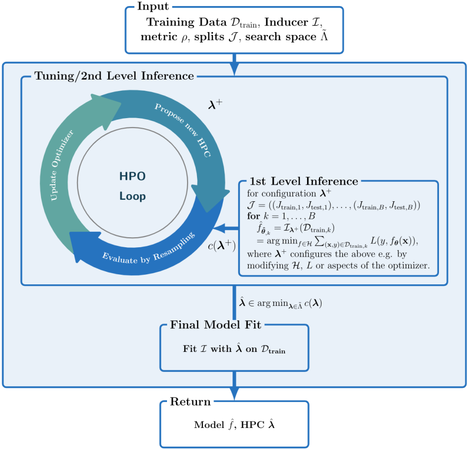






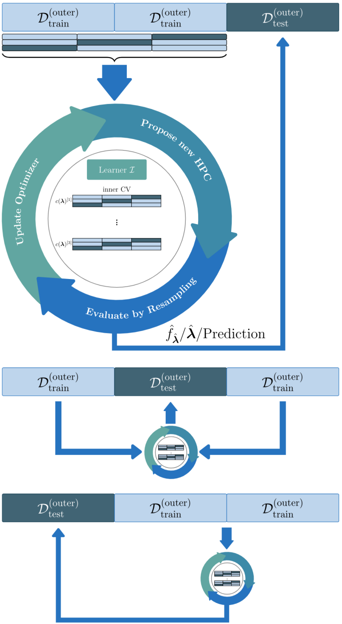
TUNING - BILEVEL INFERENCE

- Tight connection between ML and HPO
- Finding $\hat{\lambda}$ is still risk minimization w.r.t. (hyper)parameters
- **First level** / **ML:** find optimal params θ of model f w.r.t. $\mathcal{R}_{\text{emp}}$
- **Second level** / **HPO:** find optimal HPs $\hat{\lambda}$ w.r.t. $\widehat{GE}$



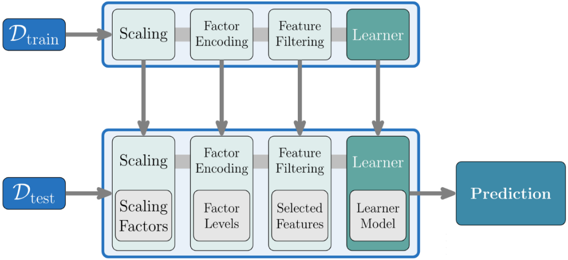
TUNING - NESTED RESAMPLING

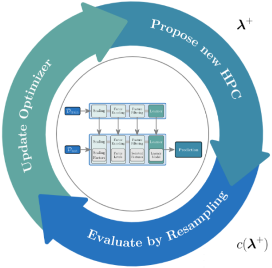
- To ensure unbiased estimation of $\widehat{GE}$, also tuned HPC $\hat{\lambda}$ need to be evaluated on an independent test set
- We need additional resampling step to prevent optimistic bias
- Combo of inner and outer resampling loop is called **nested resampling**
- Most common are train-valid-test and nested CV



PIPELINES AND AUTOML

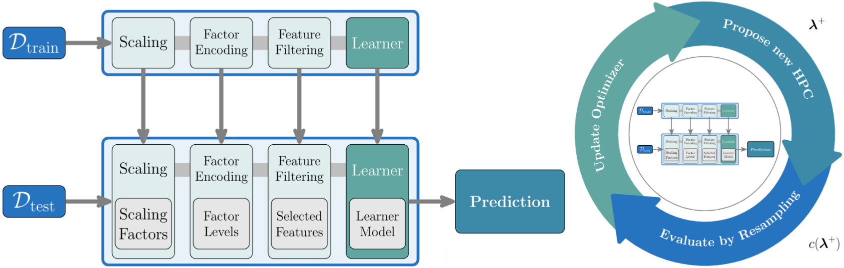
- ML typically has several data transformation steps before model fit
- If steps are in succession, data flows through linear pipeline
- NB: Each node has a train and predict step and learns params
- And usually has HPs





PIPELINES AND AUTOML

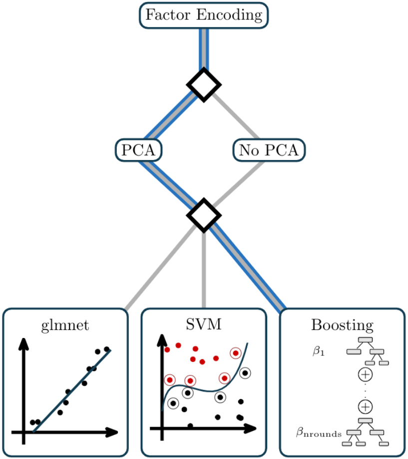
- ML typically has several data transformation steps before model fit
- If steps are in succession, data flows through linear pipeline
- NB: Each node has a train and predict step and learns params
- And usually has HPs



Pipelines are required to embed full model building into CV to avoid overfitting and biased evaluation!

PIPELINES AND AUTOML

- Further flexibility by representing pipeline as DAG
- Single source accepts $\mathcal{D}_{\text{train}}$, single sink returns predictions
- Each node represents a preprocessing operation, a learner, a postprocessing operation or controls data flow
- Can be used to implement ensembles, operator selection, ...



PIPELINES AND AUTOML

- HPs of linear pipeline are the joint set of all HPs of its contained nodes: $\tilde{\mathbf{\Lambda}} = \tilde{\mathbf{\Lambda}}_{\text{op},1} \times \cdots \times \tilde{\mathbf{\Lambda}}_{\text{op},k} \times \tilde{\mathbf{\Lambda}}_{\mathcal{I}}$
- HP space of a DAG is more complex: Depending on branching / selection different nodes and HPs are active $\rightarrow$ **hierarchical search space**

Search Space $\tilde{\Lambda}$

Name	Type	Bounds/Values	Trafo
`encoding`	C	one-hot, impact	
◇ `pca`	C	PCA, no PCA	
◇ `learner`	C	glmnet, SVM, Boosting	
`if learner = glmnet`			
`s`	R	$[-12, 12]$	2^x
`alpha`	R	$[0, 1]$	–
`if learner = SVM`			
`cost`	R	$[-12, 12]$	2^x
`gamma`	R	$[-12, 12]$	2^x
`if learner = Boosting`			
`eta`	R	$[-4, 0]$	10^x
`nrounds`	I	$\{1, \ldots, 5000\}$	–
`max_depth`	I	$\{1, \ldots, 20\}$	–

PIPELINES AND AUTOML

- HPs of linear pipeline are the joint set of all HPs of its contained nodes: $\tilde{\mathbf{\Lambda}} = \tilde{\mathbf{\Lambda}}_{\mathrm{op},1} \times \cdots \times \tilde{\mathbf{\Lambda}}_{\mathrm{op},k} \times \tilde{\mathbf{\Lambda}}_{\mathcal{I}}$
- HP space of a DAG is more complex: Depending on branching / selection different nodes and HPs are active $\rightarrow$ **hierarchical search space**

Search Space $\tilde{\Lambda}$

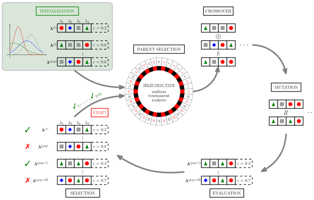
Name	Type	Bounds/Values	Trafo
`encoding`	C	one-hot, impact	
◇ `pca`	C	PCA, no PCA	
◇ `learner`	C	glmnet, SVM, Boosting	
`if learner = glmnet`			
`s`	R	$[-12, 12]$	2^x
`alpha`	R	$[0, 1]$	–
`if learner = SVM`			
`cost`	R	$[-12, 12]$	2^x
`gamma`	R	$[-12, 12]$	2^x
`if learner = Boosting`			
`eta`	R	$[-4, 0]$	10^x
`nrounds`	I	$\{1, \ldots, 5000\}$	–
`max_depth`	I	$\{1, \ldots, 20\}$	–

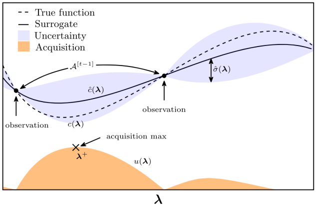
A graph that includes many preprocessing steps and learner types can be flexible enough to work on a large number of data sets

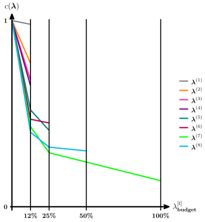
Combining such graph with an efficient tuner is key in AutoML!

HPO – MANY APPROACHES

- Evolutionary algorithms
- Bayesian / model-based optimization
- Multi-fidelity optimization, e.g. Hyperband







HPO methods can be characterized by:

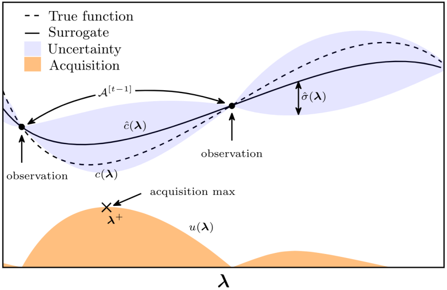
- how the exploration vs. exploitation trade-off is handled
- how the inference vs. search trade-off is handled

Further aspects: Parallelizability, local vs. global behavior, handling of noisy observations, multifidelity and search space complexity.

OPTIMIZATION – BAYESIAN OPTIMIZATION

BO sequentially iterates:

1. **Approximate $\boldsymbol{\lambda} \mapsto c(\boldsymbol{\lambda})$** by (nonlin) regression model $\hat{c}(\boldsymbol{\lambda})$, from evaluated configurations (archive / history)
2. **Propose candidates** via optimizing an acquisition function that is based on the surrogate $\hat{c}(\boldsymbol{\lambda})$
3. **Evaluate** candidate(s) proposed in 2, then go to 1

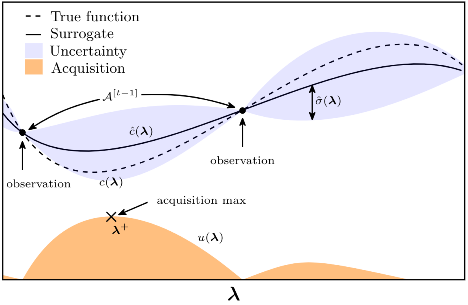


Important trade-off: **Exploration** (evaluate candidates in under-explored areas) vs. **exploitation** (search near promising areas)

OPTIMIZATION – BAYESIAN OPTIMIZATION

Surrogate Model:

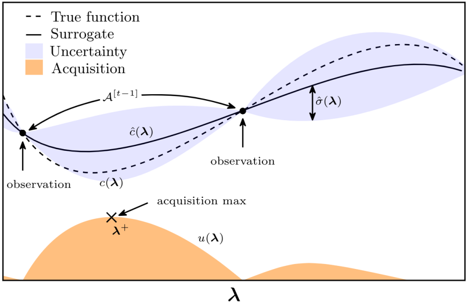
- Probabilistic modeling of $C(\boldsymbol{\lambda}) \sim (\hat{c}(\boldsymbol{\lambda}), \hat{\sigma}(\boldsymbol{\lambda}))$ with posterior mean $\hat{c}(\boldsymbol{\lambda})$ and uncertainty $\hat{\sigma}(\boldsymbol{\lambda})$.
- Typical choices for numeric spaces are Gaussian Processes; random forests for mixed spaces; Bayesian neural networks



OPTIMIZATION – BAYESIAN OPTIMIZATION

Surrogate Model:

- Probabilistic modeling of $C(\boldsymbol{\lambda}) \sim (\hat{c}(\boldsymbol{\lambda}), \hat{\sigma}(\boldsymbol{\lambda}))$ with posterior mean $\hat{c}(\boldsymbol{\lambda})$ and uncertainty $\hat{\sigma}(\boldsymbol{\lambda})$.
- Typical choices for numeric spaces are Gaussian Processes; random forests for mixed spaces; Bayesian neural networks



Acquisition Function:

- Balance exploration (high $\hat{\sigma}$) vs. exploitation (low $\hat{c}$).
- Lower confidence bound (LCB): $a(\boldsymbol{\lambda}) = \hat{c}(\boldsymbol{\lambda}) - \kappa \cdot \hat{\sigma}(\boldsymbol{\lambda})$
- Expected improvement (EI): $a(\boldsymbol{\lambda}) = \mathbb{E}\left[\max\left\{c_{\min} - C(\boldsymbol{\lambda}), 0\right\}\right]$ where ($c_{\min}$ is best cost value from archive)
- Optimizing $a(\boldsymbol{\lambda})$ is still difficult, but cheap(er)

OPTIMIZATION – FURTHER BO VARIANTS

High-dimensional and complex spaces

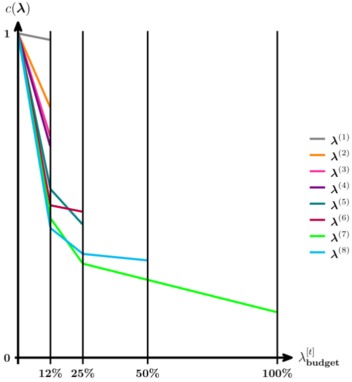
- Often, learner or pipelines are highly configurable and contain dependencies
- Fitting accurate and fast surrogates can be challenging and special surrogates may be needed (e.g., GPs with special kernels, RFs as model or BNNs with special embeddings)

Parallelization

- In standard formulation, only one point is proposed per iter and evaluated; inefficient if parallel resources are available
- Many batch proposal variants exist (batch BO)

OPTIMIZATION – SUCCESSIVE HALVING

- Races down set of HPCs to the best
- Idea: Discard bad configurations early
- Train HPCs with fraction of full budget (SGD epochs, training set size); the control parameter for this is called **multi-fidelity HP**
- Continue with better half of HPCs (w.r.t $\widehat{GE}$); with doubled budget
- Repeat until budget depleted or single HPC remains



OPTIMIZATION – HYPERBAND

Problem with SH

- Good HPCs could be killed off too early, depends on evaluation schedule

Solution: Hyperband

- Repeat SH with different start budgets $\lambda_{\text{budget}}^{[0]}$ and initial number of HPCs $p^{[0]}$
- Each SH run is called bracket
- Each bracket consumes ca. the same budget

bracket 3

t	$\lambda_{\text{budget}}^{[t]}$	$p_3^{[t]}$
0	1	8
1	2	4
2	4	2
3	8	1

bracket 2

t	$\lambda_{\text{budget}}^{[t]}$	$p_2^{[t]}$
0	2	6
1	4	3
2	8	1

bracket 1

t	$\lambda_{\text{budget}}^{[t]}$	$p_1^{[t]}$
0	4	4
1	8	2

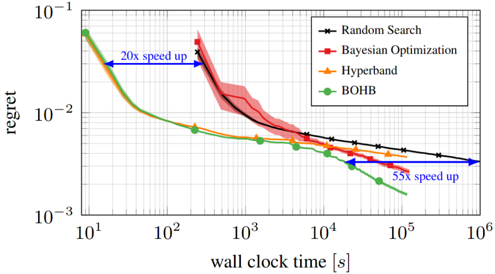
bracket 0

t	$\lambda_{\text{budget}}^{[t]}$	$p_0^{[t]}$
0	8	4

OPTIMIZATION – BOHB

Bayesian Optimization (BO) and Hyperband (HB)

- Strength of HB: Multifidelity / discard bad configs early
  - $\Rightarrow$ Most visible *early* in optimization
- Strength of BO: Sample efficiency
  - $\Rightarrow$ Most visible *later*, when initial samples result in better surrogate
- BOHB tries to combine these strengths



Optimization of six HPs of a neural network; shown is the regret (over global best known performance) of the best model found by each method at a given time.
From: Falkner et al. *BOHB: Robust and Efficient Hyperparameter Optimization at Scale*, ICML 2018

OPTIMIZATION – BOHB – ALGORITHM

General template

- Evaluates HPOs with SH as in HB
- Instead of random samples, configurations are chosen by BO
- The model is fitted to performance values of highest fidelity for which enough data is available

Point proposal with KDE

- Uses multi-dim kernel density estimator
- Divide archive into 2 groups and fit KDE on each

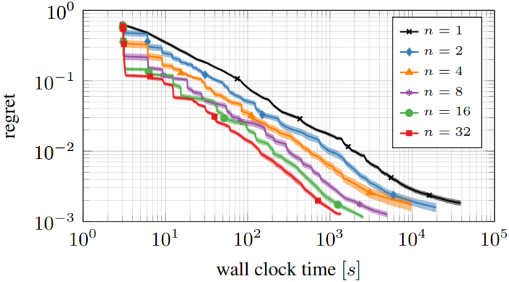
$$l(\lambda) = p(c < \alpha|\boldsymbol{\lambda}) \quad \text{('good' configurations)}$$
$$g(\lambda) = p(c \geq \alpha|\boldsymbol{\lambda}) \quad \text{('bad' configurations)}$$

(α is pre-defined percentile)

- Can show: maximizing EI is equivalent to maximizing ratio $\frac{l(\lambda)}{g(\lambda)}$

OPTIMIZATION – BOHB – VERDICT

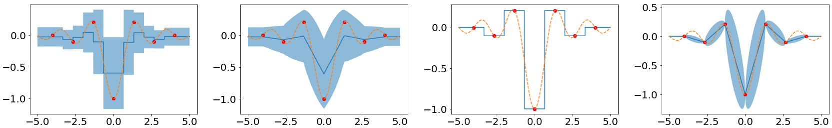
- Strong performance both early and late during optimization ("anytime performance")
- Flexible: Can be parallelized by using parallel HB methods, and noisier optimization of $\frac{l(\lambda)}{g(\lambda)}$



Performance of parallel BOHB on surrogate benchmark.
From: Falkner et al. *BOHB: Robust and Efficient Hyperparameter Optimization at Scale*, ICML 2018

OPTIMIZATION – BOHB'S SUCCESSORS: SMAC-HB

- The long-term performance heavily depends on the predictive quality of the surrogate
- Several papers indicate, other models than KDE can perform better
- SMAC-HB combines HB and BO with RFs (and GPs) as a surrogate
- On HPOBench, SMAC-HB is one of the strongest HPO approaches

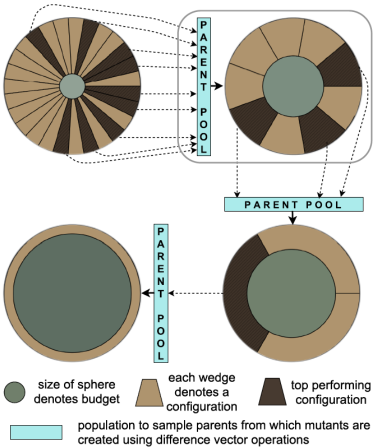


(a) w/ bootstrapping & middle splits (b) w/ bootstrapping & random splits (c) w/o bootstrapping & middle splits (d) w/o bootstrapping & random splits

Lindauer et al. *SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization*, 2021

OPTIMIZATION – BOHB'S SUCCESSORS: DEHB

- BO's overhead is often fairly large
- Evolutionary algorithms (EA) have much smaller overhead
- One of the strongest EAs is differential evolution (DE)



From: Awad et al. *DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization*, IJCAI 2021

PRACTICAL ASPECTS OF HPO

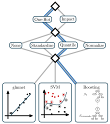
- Choosing resampling
  - No. of observations, i.i.d assumption for data sampling process

- Choosing performance measure
  - Desired implications when applying the model in practice

Name	Formula	Direction	Range	Description
Performance measures for regression				
Mean Squared Error (MSE)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$	min	$[0, \infty)$	Mean of the squared distances between the target variable y and the predicted target $\hat{y}$.
Mean Absolute Error (MAE)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left| y^{(i)} - \hat{y}^{(i)} \right|$	min	$[0, \infty)$	More robust than MSE, since it is less influenced by large errors.
R^2	$1 - \frac{\sum_{i=1}^{n_{test}} (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n_{test}} (y^{(i)} - \bar{y})^2}$	max	$(-\infty, 1]$	Compare the sum of squared errors (SSE) of the model to a constant baseline model.
Performance measures for classification based on class labels				
Accuracy (ACC)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{1}_{\{y^{(i)} = \hat{y}^{(i)}\}}$	max	$[0, 1]$	Proportion of correctly classified observations.
Balanced Accuracy (BA)	$\frac{1}{g} \sum_{k=1}^{g} \frac{1}{n_{test,k}} \sum_{y^{(i)}: y^{(i)} = k} \mathbb{1}_{\{y^{(i)} = \hat{y}^{(i)}\}}$	max	$[0, 1]$	Variant of the accuracy that accounts for imbalanced classes.
Classification Error (CE)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{1}_{\{y^{(i)} \neq \hat{y}^{(i)}\}}$	min	$[0, 1]$	$\mathrm{CE} = 1 - \mathrm{ACC}$ is the proportion of incorrect predictions.
ROC measures	$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	max	$[0, 1]$	True Positive Rate: how many observations of the positive class 1 are predicted as 1?
	$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	min	$[0, 1]$	False Positive Rate: how many observations of the negative class 0 are falsely predicted as 1?
	$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	max	$[0, 1]$	True Negative Rate: how many observations of the negative class 0 are predicted as 0?
	$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}$	min	$[0, 1]$	False Negative Rate: how many observations of the positive class 1 were falsely predicted as 0?
	$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	max	$[0, 1]$	Positive Predictive Value: how likely is a predicted 1 a true 1?
	$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{FN}+\mathrm{TN}}$	max	$[0, 1]$	Negative Predictive Value: how likely is a predicted 0 a true 0?
F_1	$2\frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}}$	max	$[0, 1]$	F_1 is the harmonic mean of PPV and TPR. Especially useful for imbalanced classes.
Cost measure	$\sum_{i=1}^{n_{test}} C(y^{(i)}, \hat{y}^{(i)})$	min	$[0, \infty)$	Cost of incorrect predictions based on a (usually non-negative) cost matrix $C \in \mathbb{R}^{g \times g}$.
Performance measures for classification based on class probabilities				
Brier Score (BS)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \sum_{k=1}^{g} \left(\hat{\pi}_k(\mathbf{x}^{(i)}) - \sigma_k(y^{(i)}) \right)^2$	min	$[0, 1]$	Measures squared distances of probabilities from the one-hot encoded class labels.
Log-Loss (LL)	$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(-\sum_{k=1}^{g} \sigma_k(y^{(i)}) \log(\hat{\pi}_k(\mathbf{x}^{(i)})) \right)$	min	$[0, \infty)$	A.k.a. Bernoulli, binomial or cross-entropy loss
AUC		max	$[0, 1]$	Area under the ROC curve.

$\hat{y}^{(i)}$ denotes the predicted label for observation $\mathbf{x}^{(i)}$. ACC, BA, CE, BS, and LL can be used for multi-class classification with g classes. For AUC multiclass extensions exist as well. The notation $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function. $\sigma_k(y) = \mathbb{1}_{\{y=k\}}$ is 1 if y is class k, 0 otherwise (multi-class one-hot encoding). $n_{test,k}$ is the number of observations in the test set with class k. $\hat{\pi}_k(\mathbf{x})$ is the estimated probability for observation $\mathbf{x}^{(i)}$ of belonging to class k. TP is the number of true positives (observations of class 1 with predicted class 1), FP is the number of false positives (observations of class 0 with predicted class 1), TN is the number of true negatives (observations of class 0 with predicted class 0), and FN is the number of false negatives (observations of class 1 with predicted class 0).

- Choosing a pipeline and search space
  - Numeric HPs of arbitrary size should be tuned on log scale
  - Size of search space results in different trade-offs:
    too small may miss out well performing HPCs;
    too large makes optimization more difficult



Search Space $\tilde{\Lambda}$

Name	Type	Bounds/Values	Trafo
◇ `encoding`	C	One-Hot, Impact	
◇ `scaling`	C	None, Standardize, Normalize, Quantile Transformation	
◇ `learner`	C	glmnet, SVM, Boosting	
`if learner = glmnet`			
`s`	R	$[-12, 12]$	2^x
`alpha`	R	$[0, 1]$	–
`if learner = SVM`			
`cost`	R	$[-12, 12]$	2^x
`gamma`	R	$[-12, 12]$	2^x
`if learner = Boosting`			
`eta`	R	$[-4, 0]$	10^x
`nrounds`	I	$\{1, \ldots, 5000\}$	–
`max_depth`	I	$\{1, \ldots, 20\}$	–

PRACTICAL ASPECTS OF HPO

- Choosing HPO algorithm
  - For few HPS (1-2), grid search could be used for having a controlled study (but is not recommended efficiency-wise)
  - BO with GPs for up to 10 numeric HPs
  - BO with RFs handle mixed HP spaces
  - Random search and Hyperband work well as long as the “effective” dimension is low
  - EAs are somewhat in-between BO and RS, can handle very complex spaces, but less sample efficient than BO
  - **Also: use something that’s stable and robust! More an aspect of the implementation than the algo!**

- When to terminate HPO
  - Specify a certain amount of runtime/budget beforehand
  - Set a lower bound regarding $\widehat{GE}$
  - Terminate if performance improvement stagnates
  - Terminate if acquisition function values reach a threshold (BO)

PRACTICAL ASPECTS OF HPO

- Warm starts
  - Evaluations (e.g., weight sharing of neural networks)
  - Optimization (intializing with HPCs that worked well before)

- Control of execution
  - Parallelizability of HPO algorithms differs strongly
  - HPO execution can be parallelized at different levels (outer resampling, iteration, evaluation, inner resampling, model fit)

More on practical aspects $\rightarrow$ Bischl, ..., Lindauer. *Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges*, under review, 2021

WHAT DOES ACTUALLY WORK?

Problem:

- New HPO methods are proposed frequently
- Benchmarking new methods and SOTA algorithms is expensive: Papers often only use toy problems, synthetic functions or a very limited number of real world problems
  $\rightarrow$ No clear indication of what really works in practice!

Solution:

- Easy to use and reproducible HPO benchmark suites with **practically relevant** problems for comparison of HPO methods

HPO BENCHMARK SUITES

HPOBench [Eggensperger et al. NeurIPS'21 Datasets and Benchmarks Track]

- Successor of HPOlib
- Collection of 12 benchmark families; in total > 100 HPO problems
- Mix of tabular, surrogate and real benchmark problems
- Also allows for benchmarking multifidelity HPO methods
- Benchmarks are containerized making them easily reproducible

YAHPO Gym [Pfisterer, Schneider et al. 2021]

- Collection of 9 benchmark families constituting over 700 multifidelity multicriteria HPO problems
- Surrogate benchmarks using neural-network based instance surrogates
- fast inference ($<$ 50 ms) & low memory footprint ($\sim$ 5 MB)

Others: HPO-B [Arango et al. NeurIPS'21 Datasets and Benchmarks Track]

SYSTEMATIC AUTOML BENCHMARK

What is it?

- Open source framework for benchmarking state-of-the-art AutoML systems using OpenML datasets

Why?

- Fair and easy-to-use comparison
  $\rightarrow$ open source – open science

Who is involved?

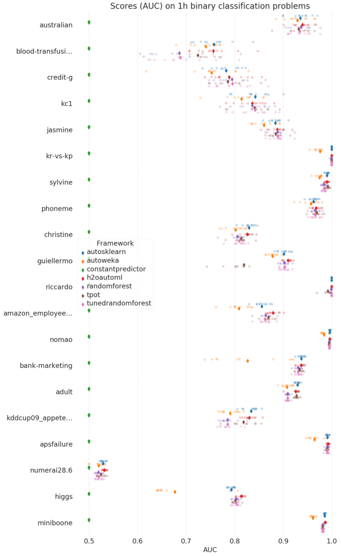
OpenML

LMU

H_2O AutoML

Some Conclusions:

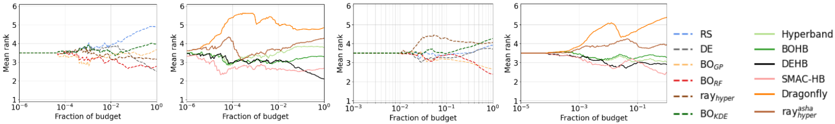
- No AutoML system consistently outperforms others
- Tuned random forest is very competitive



Gijsbers et al. *An Open Source AutoML Benchmark*, AutoML WS at ICML 2019

AUTOML – CHALLENGES

- Most efficient HPO approach? Good benchmarks often missing (but things are slowly changing for the better)



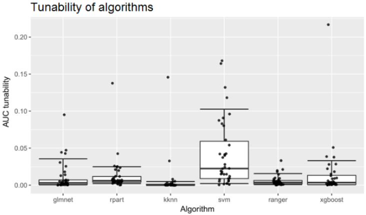
Mean rank-over-time across 32 repetitions for black-box and multi-fidelity optimizers.

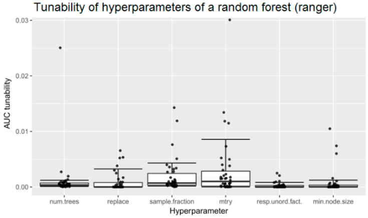
Eggensperger et al. *HPOBench*, NeurIPS'21 Datasets and Benchmarks Track

- How to integrate human a-priori knowledge?
- How can we best (computationally) transfer "experience" into AutoML? Warmstarts, learned search spaces, etc.
- Multi-objective goals, including model intepretability
- AutoML as a process is too much of a black-box, hurts adoption

IMPORTANCE OF HYPERPARAMETERS

- For users very often unclear, what to tune and how to setup optimization.
- Addresses problem of HP importance, optimal defaults and empirical design of search spaces
- Theoretical definitions for above quantities; computation from large ML databases and aggregate of surrogates





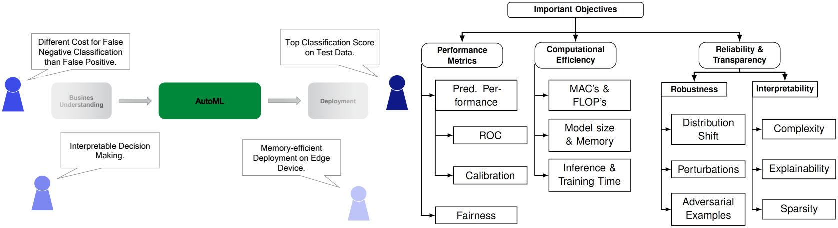
Tunability of an algorithm S_k:
$d_k := S_k(\theta_{def}) - S_k(\theta_k^\star)$

Tunability of a parameter $\theta^{(}i)$:
$d_k^{(i)} := S_k(\theta_{def}) - S_k(\theta_k^{(i)\star})$

Probst, Boulesteix, Bischl. *Tunability: Importance of Hyperparameters of Machine Learning Algorithms*, JMLR 2019
van Rijn, Hutter. *Hyperparameter Importance Across Datasets*, KDD 2018

MANY STAKEHOLDERS - MULTIPLE OBJECTIVES

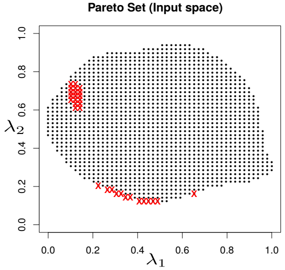
- Optimizing a model only for prediction is often unrealistic.



- Some tasks can't be distilled into one single metric.
- Stakeholders in ML process like different properties of model.

Horn, Bischl et al. *Multi-objective parameter configuration of machine learning algorithms using model-based optimization*, 2016 IEEE symposium series on computational intelligence

MULTI-OBJ. HYPERPARAMETER OPTIMIZATION



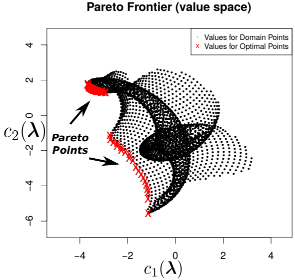


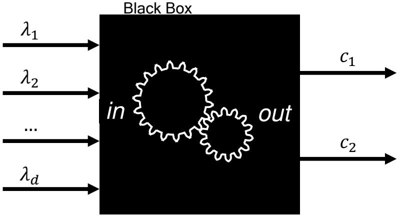
Image source: Daniel Hernández-Lobato

- $\mathbf{c}(\boldsymbol{\lambda}) = (c_1(\boldsymbol{\lambda}), \dots, c_m(\boldsymbol{\lambda}))$
- $\boldsymbol{\lambda}$ *dominates* $\tilde{\boldsymbol{\lambda}}$ if

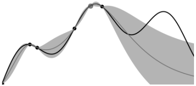
$$\forall i \in \{1, \dots, m\} : c_i(\boldsymbol{\lambda}) \leq c_i(\tilde{\boldsymbol{\lambda}})$$
$$\text{and } \exists i \in \{1, \dots, m\} : c_i(\boldsymbol{\lambda}) < c_i(\tilde{\boldsymbol{\lambda}})$$

- Set of non-dominated solutions:

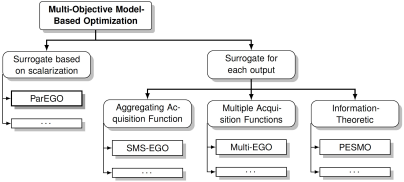
$$\Lambda^* := \{\boldsymbol{\lambda} \in \tilde{\boldsymbol{\Lambda}} | \nexists \tilde{\boldsymbol{\lambda}} \in \tilde{\boldsymbol{\Lambda}} : \tilde{\boldsymbol{\lambda}} \text{ dominates } \boldsymbol{\lambda}\}$$



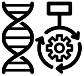
POPULAR METHODS FOR MULTI-OBJ. AUTOML



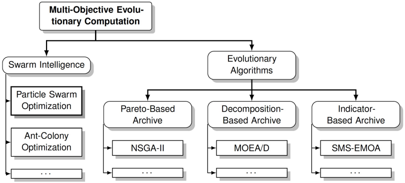
Model-Based Optimization



Tends to be more sample efficient.



Evolutionary Algorithms

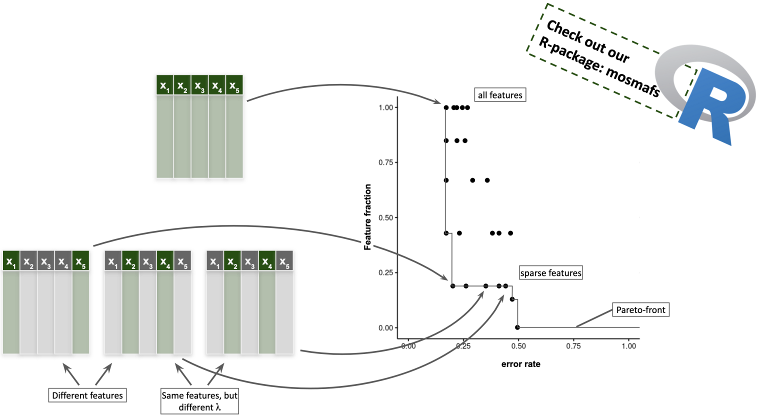


Works well with complex and awkward search spaces.

Horn, Bischl et al. *Model-based multi-objective optimization: taxonomy, multi-point proposal, toolbox and benchmark*, International Conference on Evolutionary Multi-Criterion Optimization 2015

MULTI-OBJ. FEATURE SELECTION AND HPO

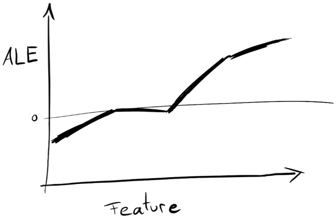
- Should we first select features or first optimize hyperparameters?
  $\longrightarrow$ Do both simultaneously!



Binder, Moosbauer, Bischl et al. *Multi-objective hyperparameter tuning and feature selection using filter ensembles*, GECCO 2020

QUANTIFYING COMPLEXITY FOR IML

- Minimizing model complexity maximizes interpretability
- Measure model complexity based on FANOVA in model-agnostic way: number of features, interaction strength, main effect complexity
- Enables multi-objective model-selection for interpretability



FANOVA Decomposition:

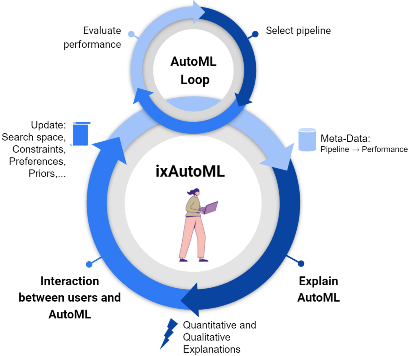
$$f(x) = \overbrace{f_0}^{\text{Intercept}} + \overbrace{\sum_{j=1}^{p} f_j(x_j)}^{\text{1st order effects}} + \overbrace{\sum_{S \subseteq \{1,\ldots,p\}, |S| \geq 2} f_S(x_S)}^{\text{Higher order effects}}$$

Molnar, et al. *Quantifying Interpretability of Arbitrary ML Models Through Functional Decomposition*, ECML 2019

HUMAN-CENTERED AUTOML

- Fully automated ML design can also receive pushback:
  - How to verify results (i.e., ML pipelines)?
  - How to bring in human expertise?
  - How to integrate into prototype-driven workflows?

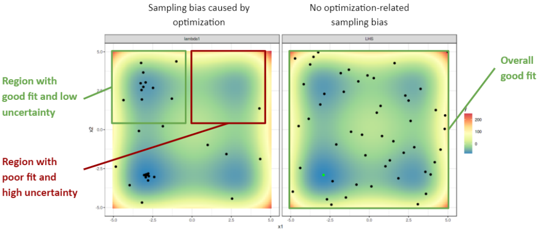
⇝ Human-centered AutoML instead of fully automated ML?

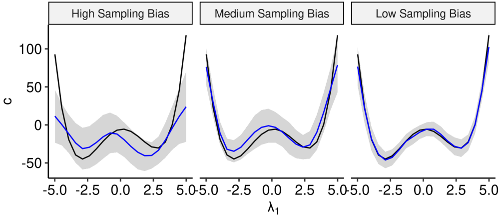


EXPLAINABLE HPO

Goal: Explain HPO via Partial Dependence Plots (PDPs)

- Problem: Optimization traces are not *iid* samples
- Low reliability / large uncertainties in the PDP estimation
- Can't (easily) change sampling behavior (optimization)

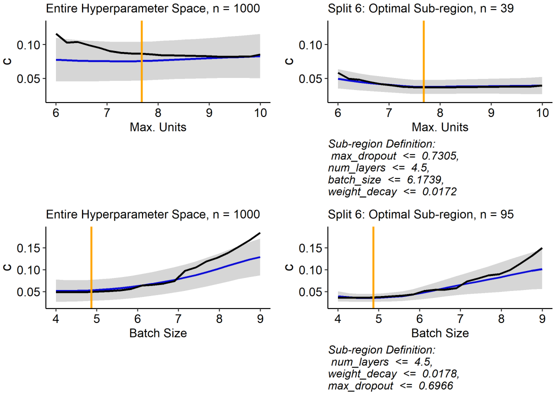




Black: true function;
Blue: PDP; Grey area: uncertainty

EXPLAINABLE HPO

Solution: PDP uncertainty measure + recursively partition search space to get low-variance explanations in interesting areas



- Loss (variance across projected dimension):
  $$L\left(\boldsymbol{\lambda}_S, \mathcal{N}'\right) = \sum_{i \in \mathcal{N}} \left(\hat{s}^2\left(\boldsymbol{\lambda}_S, \boldsymbol{\lambda}_C^{(i)}\right) - \hat{s}^2_{S|\mathcal{N}'}\left(\boldsymbol{\lambda}_S\right)\right)^2$$
- Splitting criterion:
  $$\mathcal{R}_{L2}(\mathcal{N}') = \sum_{g=1}^{G} L(\boldsymbol{\lambda}_S^{(g)}, \mathcal{N}')$$

Moosbauer et al. *Explaining HPO via Partial Dependence Plots*, NeurIPS'21

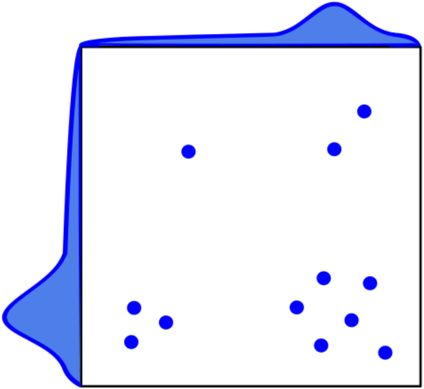
INTEGRATING HUMAN A-PRIORI KNOWLEDGE

- ML practitioners often have an intuition for promising hyperparameter configurations

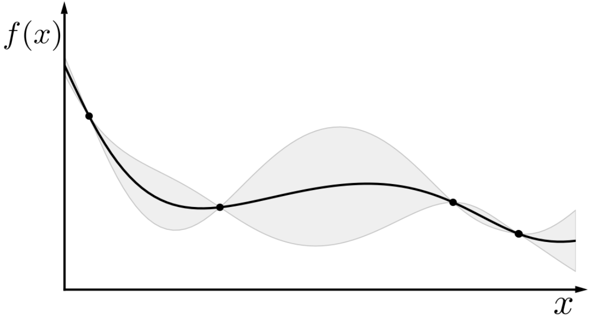
⇝ Sampling of configurations should focus in these regions

- However, practitioners can also be wrong with their intuition

⇝ Over time, we should trust the evaluated configurations and the surrogate more than the human expert

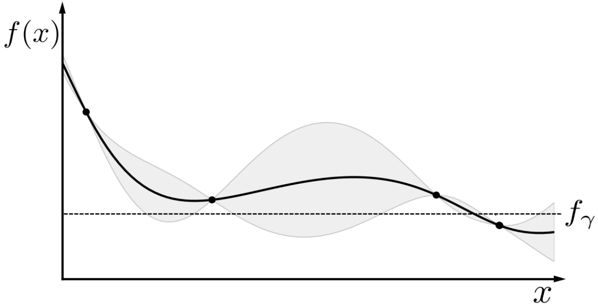


HUMAN A-PRIORI KNOWLEDGE – BOPRO



- Instead of modelling $f(x)$ (a.k.a. $c(\boldsymbol{\lambda})$), we model whether a configuration is "good" or "bad" (see KDE)

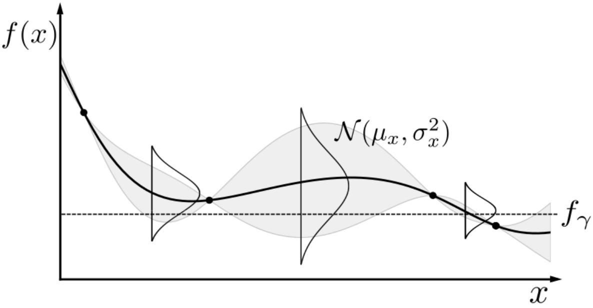
HUMAN A-PRIORI KNOWLEDGE – BOPRO



- Instead of modelling $f(x)$ (a.k.a. $c(\boldsymbol{\lambda})$), we model whether a configuration is "good" or "bad" (see KDE)

Souza et al. *Bayesian Optimization with a Prior for the Optimum*, ECML 2021

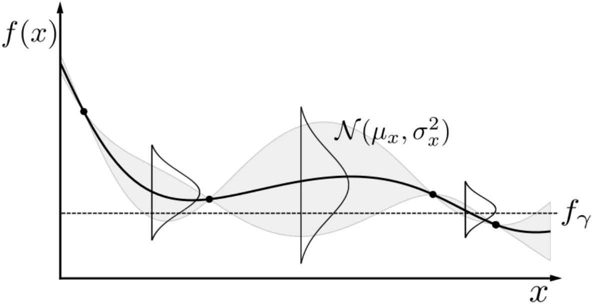
HUMAN A-PRIORI KNOWLEDGE – BOPRO



- Instead of modelling $f(x)$ (a.k.a. $c(\boldsymbol{\lambda})$), we model whether a configuration is "good" or "bad" (see KDE)
- Using the Gaussian distribution of a GP, we can determine the probability of being good $\hat{c}_g$

Souza et al. *Bayesian Optimization with a Prior for the Optimum*, ECML 2021

HUMAN A-PRIORI KNOWLEDGE – BOPRO

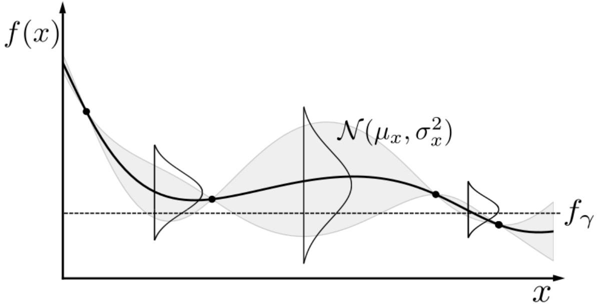


- Instead of modelling $f(x)$ (a.k.a. $c(\boldsymbol{\lambda})$), we model whether a configuration is "good" or "bad" (see KDE)
- Using the Gaussian distribution of a GP, we can determine the probability of being good $\hat{c}_g$
- Combine with human prior on being a promising configuration P_g

$$g(\boldsymbol{\lambda}) \propto P_g(\boldsymbol{\lambda}) \hat{c}_g(\boldsymbol{\lambda})^{\frac{t}{\beta}}$$

Souza et al. *Bayesian Optimization with a Prior for the Optimum*, ECML 2021

HUMAN A-PRIORI KNOWLEDGE – BOPRO



- Instead of modelling $f(x)$ (a.k.a. $c(\boldsymbol{\lambda})$), we model whether a configuration is "good" or "bad" (see KDE)
- Using the Gaussian distribution of a GP, we can determine the probability of being good $\hat{c}_g$
- Combine with human prior on being a promising configuration P_g

$$g(\boldsymbol{\lambda}) \propto P_g(\boldsymbol{\lambda}) \hat{c}_g(\boldsymbol{\lambda})^{\frac{t}{\beta}}$$

- Over time t, the influence of the human prior gets weaker

Souza et al. *Bayesian Optimization with a Prior for the Optimum*, ECML 2021

HUMAN A-PRIORI KNOWLEDGE – πBO

- BOPro has several assumptions on how to model the observations
- πBO is simpler: augment the the acquisition function of BO by a human prior preference

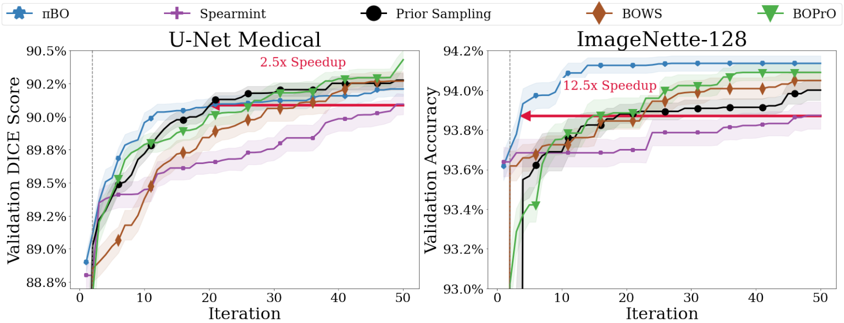
$$a_\pi(\boldsymbol{\lambda}) = a(\boldsymbol{\lambda})\pi(\boldsymbol{\lambda})^{\frac{\beta}{t}}$$

Advantages of πBO:

- Can be combined with any acquisition function
- Same convergence guarantees as with the original acquisition functions (e.g., EI)
- Can again recover from misleading a-priori knowledge

Hvarfner et al. *πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization*, under review, 2021

HUMAN A-PRIORI KNOWLEDGE – πBO



Hvarfner et al. π*BO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization*, under review, 2021

CONCLUSIONS

- Sample efficiency is key!
- Can be achieved by different sampling or evaluation strategies
- Multi-objective HPO (/AutoML) is important in practice
- Never forget the human in the loop!

CONCLUSIONS

- Sample efficiency is key!
- Can be achieved by different sampling or evaluation strategies
- Multi-objective HPO (/AutoML) is important in practice
- Never forget the human in the loop!

Have Fun at the AutoML Fall School!