AutoML: Hyperparameter Optimization

Wrap Up

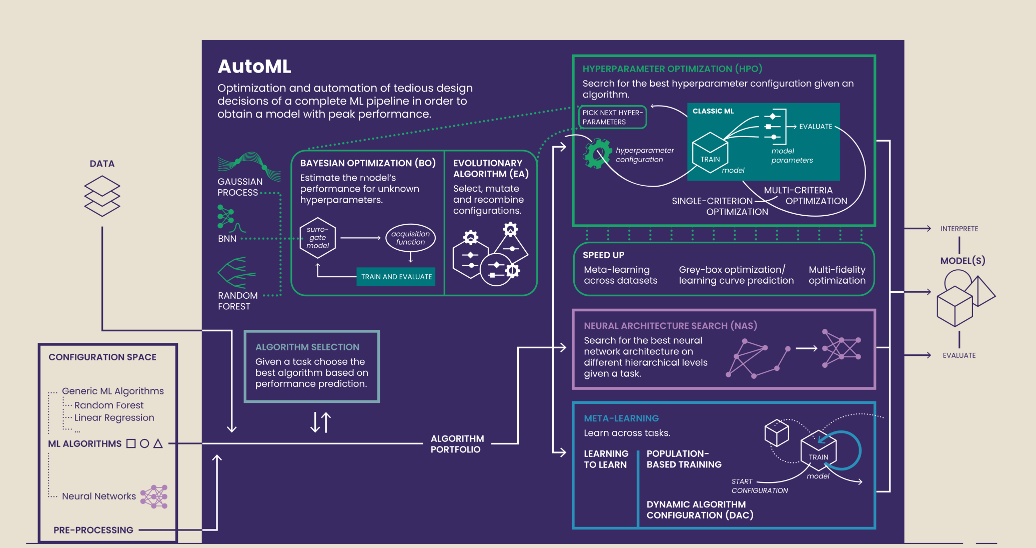
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer

From HPO to AutoML

So far we covered

- Mechanisms to select ML algorithms for a data set (algorithm selection)
- HPO as black-box optimization
  - Grid- and random search, EAs, BO
- HPO as a grey box problem
  - Hyperband, BOHB
- Neural Architecture Search (NAS)
  - One-Shot approaches, DART
- Dynamic algorithm configuration (learning to learn)
  - Adapt configuration during training

From HPO to AutoML



What is missing?

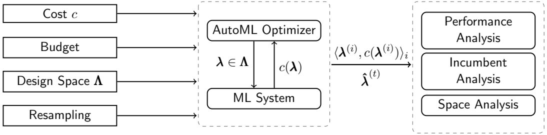
What do I need to know as an AutoML user?

- ~~Nothing, because it is automatic.~~
- Understand limitations of AutoML and framework.
- Know how to interpret the results.
- Maybe: Preprocessing and feature extraction.

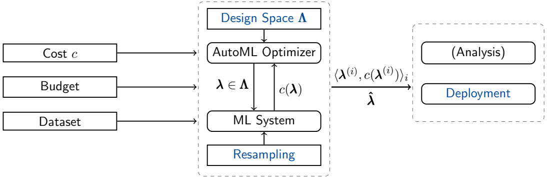
Ingredients to implement an AutoML?

- HPO algorithm
- ML / Pipeline framework
- Parallelization / Multifidelity
- Process encapsulation and time capping

Academic view:



Practitioners view:



Choice of Learning Algorithm

- A plethora of learners exists, for different data sets different models are likely needed.
- Studies and experience show:
  One these is often good – on tabular data:
  - penalized regression
  - SVM
  - gradient boosting
  - random forests
  - (neural networks)
- Example: Auto-Sklearn 2.0 [Feurer et al. 2020] uses:
  - extra trees
  - gradient boosting
  - passive aggressive
  - random forest
  - linear model

Choice of Search Space for a Learning Algorithm

Algorithm	Hyperparameter	Type	Lower	Upper	Trafo
glmnet					
(Elastic net)	alpha	numeric	0	1	-
	lambda	numeric	-10	10	2^x
rpart					
(Decision tree)	cp	numeric	0	1	-
	maxdepth	integer	1	30	-
	minbucket	integer	1	60	-
	minsplit	integer	1	60	-
kknn					
(k-nearest neighbor)	k	integer	1	30	-
svm					
(Support vector machine)	kernel	discrete	-	-	-
	cost	numeric	-10	10	2^x
	gamma	numeric	-10	10	2^x
	degree	integer	2	5	-
ranger					
(Random forest)	num.trees	integer	1	2000	-
	replace	logical	-	-	-
	sample.fraction	numeric	0.1	1	-
	mtry	numeric	0	1	$x \cdot p$
	respect.unordered.factors	logical	-	-	-
	min.node.size	numeric	0	1	n^x
xgboost					
(Gradient boosting)	nrounds	integer	1	5000	-
	eta	numeric	-10	0	2^x
	subsample	numeric	0.1	1	-
	booster	discrete	-	-	-
	max_depth	integer	1	15	-
	min_child_weight	numeric	0	7	2^x
	colsample_bytree	numeric	0	1	-
	colsample_bylevel	numeric	0	1	-
	lambda	numeric	-10	10	2^x
	alpha	numeric	-10	10	2^x

Taken from [Probst et al. 2019].

Ranges often selected based on experience

- See other AutoML frameworks: e.g. Auto-Sklearn 2.0 [Feurer et al. 2020]
- Sensitivity analysis often does not exist for ML algorithms
- Check literature on specific ML algorithm

Choice of Search Space for a Learning Algorithm

Parameter	Def.P	Def.O	Tun.P	Tun.O	$q_{0.05}$	$q_{0.95}$
glmnet			0.069	0.024		
alpha	1	0.403	0.038	0.006	0.009	0.981
lambda	0	0.004	0.034	0.021	0.001	0.147
rpart			0.038	0.012		
cp	0.01	0	0.025	0.002	0	0.008
maxdepth	30	21	0.004	0.002	12.1	27
minbucket	7	12	0.005	0.006	3.85	41.6
minsplit	20	24	0.004	0.004	5	49.15
kknn			0.031	0.006		
k	7	30	0.031	0.006	9.95	30
svm			0.056	0.042		
kernel	radial	radial	0.030	0.024		
cost	1	682.478	0.016	0.006	0.002	920.582
gamma	$1/p$	0.005	0.030	0.022	0.003	18.195
degree	3	3	0.008	0.014	2	4
ranger			0.010	0.006		
num.trees	500	983	0.001	0.001	206.35	1740.15
replace	TRUE	FALSE	0.002	0.001		
sample.fraction	1	0.703	0.004	0.002	0.323	0.974
mtry	$\sqrt{p}$	$p \cdot 0.257$	0.006	0.003	0.035	0.692
respect.unordered.factors	TRUE	FALSE	0.000	0.000		
min.node.size	1	1	0.001	0.001	0.007	0.513
xgboost			0.043	0.014		
nrounds	500	4168	0.004	0.002	920.7	4550.95
eta	0.3	0.018	0.006	0.005	0.002	0.355
subsample	1	0.839	0.004	0.002	0.545	0.958
booster	gbtree	gbtree	0.015	0.008		
max_depth	6	13	0.001	0.001	5.6	14
min_child_weight	1	2.06	0.008	0.002	1.295	6.984
colsample_bytree	1	0.752	0.006	0.001	0.419	0.864
colsample_bylevel	1	0.585	0.008	0.001	0.335	0.886
lambda	1	0.982	0.003	0.002	0.008	29.755
alpha	1	1.113	0.003	0.002	0.002	6.105

Table 3: Defaults (package defaults (Def.P) and optimal defaults (Def.O)), tunability of the hyperparameters with the package defaults (Tun.P) and our optimal defaults (Tun.O) as reference and tuning space quantiles ($q_{0.05}$ and $q_{0.95}$) for different parameters of the algorithms.

Taken from [Probst et al. 2019].

Ranges often selected based on experience

- See other AutoML frameworks: e.g. Auto-Sklearn 2.0 [Feurer et al. 2020]
- Sensitivity analysis often does not exist for ML algorithms
- Check literature on specific ML algorithm

Options for automation:

1. Use huge search space to cover all possibilities (combine with meta-learning for good initial design for Bayesian optimization)
   - Use results of meta-experiments to obtain smaller search space that is estimated to work well.

Choice of Search Space for a Learning Algorithm

Parameter	Def.P	Def.O	Tun.P	Tun.O	$q_{0.05}$	$q_{0.95}$
glmnet			0.069	0.024		
alpha	1	0.403	0.038	0.006	0.009	0.981
lambda	0	0.004	0.034	0.021	0.001	0.147
rpart			0.038	0.012		
cp	0.01	0	0.025	0.002	0	0.008
maxdepth	30	21	0.004	0.002	12.1	27
minbucket	7	12	0.005	0.006	3.85	41.6
minsplit	20	24	0.004	0.004	5	49.15
kknn			0.031	0.006		
k	7	30	0.031	0.006	9.95	30
svm			0.056	0.042		
kernel	radial	radial	0.030	0.024		
cost	1	682.478	0.016	0.006	0.002	920.582
gamma	$1/p$	0.005	0.030	0.022	0.003	18.195
degree	3	3	0.008	0.014	2	4
ranger			0.010	0.006		
num.trees	500	983	0.001	0.001	206.35	1740.15
replace	TRUE	FALSE	0.002	0.001		
sample.fraction	1	0.703	0.004	0.002	0.323	0.974
mtry	$\sqrt{p}$	$p \cdot 0.257$	0.006	0.003	0.035	0.692
respect.unordered.factors	TRUE	FALSE	0.000	0.000		
min.node.size	1	1	0.001	0.001	0.007	0.513
xgboost			0.043	0.014		
nrounds	500	4168	0.004	0.002	920.7	4550.95
eta	0.3	0.018	0.006	0.005	0.002	0.355
subsample	1	0.839	0.004	0.002	0.545	0.958
booster	gbtree	gbtree	0.015	0.008		
max_depth	6	13	0.001	0.001	5.6	14
min_child_weight	1	2.06	0.008	0.002	1.295	6.984
colsample_bytree	1	0.752	0.006	0.001	0.419	0.864
colsample_bylevel	1	0.585	0.008	0.001	0.335	0.886
lambda	1	0.982	0.003	0.002	0.008	29.755
alpha	1	1.113	0.003	0.002	0.002	6.105

Table 3: Defaults (package defaults (Def.P) and optimal defaults (Def.O)), tunability of the hyperparameters with the package defaults (Tun.P) and our optimal defaults (Tun.O) as reference and tuning space quantiles ($q_{0.05}$ and $q_{0.95}$) for different parameters of the algorithms.

Taken from [Probst et al. 2019].

Ranges often selected based on experience

- See other AutoML frameworks: e.g. Auto-Sklearn 2.0 [Feurer et al. 2020]
- Sensitivity analysis often does not exist for ML algorithms
- Check literature on specific ML algorithm

Options for automation:

1. Use huge search space to cover all possibilities (combine with meta-learning for good initial design for Bayesian optimization)
    - Use results of meta-experiments to obtain smaller search space that is estimated to work well.
2. Start with a small space and increase bit by bit

Choice of Resampling Strategy

For computation of generalization error / cost:

$$c(\boldsymbol{\lambda}) = \frac{1}{k}\sum_{i=1}^{k}\widehat{GE}_{\mathcal{D}^i_{\text{val}}}\left(\mathcal{I}(\mathcal{D}^i_{\text{train}}, \boldsymbol{\lambda})\right)$$

Rules of thumb:

- Default: 10-fold CV ($k = 10$)
- Huge datasets: holdout
- Tiny datasets: 10x10 repeated CV
- Stratification for imbalanced classes

Choice of Resampling Strategy

For computation of generalization error / cost:

$$c(\boldsymbol{\lambda}) = \frac{1}{k} \sum_{i=1}^{k} \widehat{GE}_{\mathcal{D}^i_{\text{val}}} \left(\mathcal{I}(\mathcal{D}^i_{\text{train}}, \boldsymbol{\lambda})\right)$$

Rules of thumb:

- Default: 10-fold CV ($k = 10$)
- Huge datasets: holdout
- Tiny datasets: 10x10 repeated CV
- Stratification for imbalanced classes

Watch out for this:

- Small sample size because of imbalances
- Leave-one-object out
- Time dependencies
- A good AutoML system should let you customize resampling
- Meta-learn good resampling strategy [Feurer et al. 2020]

Choice of Optimization Algorithm

Choose optimization algorithm based on ...

- complexity of search space / budget
- time-costs of evaluations

Complex search space

$\rightarrow$ BO with RF, EA with exploratory character, TPE,

[1]Still has its own hyperparameters[Lindauer et al. 2019]

Choice of Optimization Algorithm

Choose optimization algorithm based on . . .

- complexity of search space / budget
- time-costs of evaluations

Complex search space

$\rightarrow$ BO with RF, EA with exploratory character, TPE,

Numerical (lower-dim) search space and tight budget

$\rightarrow$ BO with GP[1]

[1] Still has its own hyperparameters[Lindauer et al. 2019]

Choice of Optimization Algorithm

Choose optimization algorithm based on ...

- complexity of search space / budget
- time-costs of evaluations

Complex search space

$\rightarrow$ BO with RF, EA with exploratory character, TPE,

Numerical (lower-dim) search space and tight budget

$\rightarrow$ BO with GP[1]

Expensive evaluations

$\rightarrow$ Hyperband, BOHB

[1] Still has its own hyperparameters[Lindauer et al. 2019]

Choice of Optimization Algorithm

Choose optimization algorithm based on ...

- complexity of search space / budget
- time-costs of evaluations

Complex search space

$\rightarrow$ BO with RF, EA with exploratory character, TPE,

Numerical (lower-dim) search space and tight budget

$\rightarrow$ BO with GP[1]

Expensive evaluations

$\rightarrow$ Hyperband, BOHB

Deep learning

$\rightarrow$ common practice: Parameterize architectures, then HPO – better do it jointly!

$\rightarrow$ one-shot models and gradient-based optimization

[1] Still has its own hyperparameters [Lindauer et al. 2019]