

AutoML: Hyperparameter Optimization

Example and Practical Hints

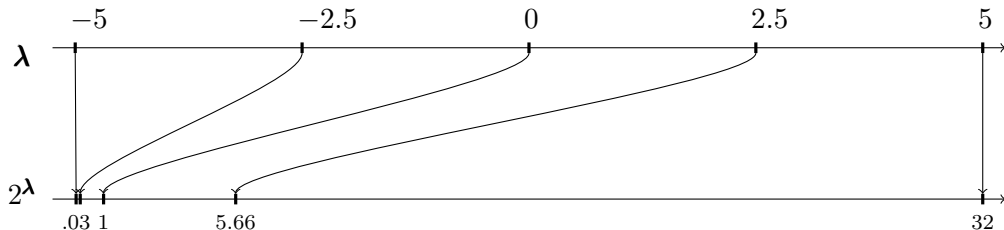
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Scaling and Ranges I

- Knowledge about hyperparameters can help to guide the optimization
- E.g., it can be beneficial to optimize hyperparameters on a non-uniform scale.

Example: regularization hyperparameter on log-scale

- Many ML algorithms have non-neg. hyperparameters (e.g. regularization constant), for which it can make sense to try out very small and very large values during tuning
- Usual trick: put on log-scale: C of SVM: $[2^{-15}, 2^{15}] = [0.00003, 32768]$



Scaling and Ranges II

- Similar to the scale, e.g., linear or logarithmic, upper and lower bounds for hyperparameters have to be specified as many optimizers require them
- Setting these correctly usually requires deeper knowledge about the inner workings of the ML method and a lot of practical experience
- Furthermore, if $\hat{\lambda}$ is close to the border of Λ the ranges should be increased (or a different scale should be selected), but many algorithms do not do this automatically
- Meta-Learning can help to decide which hyperparameters should be tuned in which ranges

Default Heuristics

- Instead of using static defaults, we sometimes can compute hyperparameter defaults heuristically, based on simple data characteristics.
- A lot faster than tuning and sometimes can work well, although not many guarantees exists and it is often unclear how well this works across many different data scenarios
- Well-known example: Number of random features to consider for splitting in random forest: $m_{\text{try}} = \sqrt{p}$, where p is total number of features.
- For the RBF-SVM a data dependent default for the γ parameter (inverse kernel width) can be computed by using the (inverse of the) median of the pairwise distances $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ between data points (or a smaller random subset for efficiency)

Tuning Example - Setup

We want to train a *spam detector* on the popular Spam dataset¹.

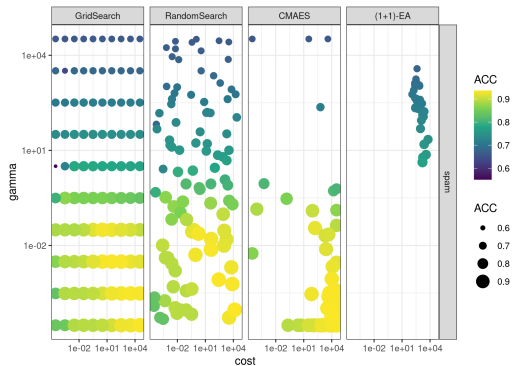
- The learning algorithm is a support vector machine (SVM) with RBF kernel.
- The hyperparameters we optimize are
 - ▶ Cost parameter $\text{cost} \in [2^{-15}, 2^{15}]$.
 - ▶ Kernel parameter $\gamma \in [2^{-15}, 2^{15}]$.
- We compare four different optimizers
 - ▶ Random search
 - ▶ Grid search
 - ▶ A $(1 + 1)$ -selection EA and Gauss mutation with $\sigma = 1$.
 - ▶ *CMAES* - efficient EA that generates offspring from a multivariate Gaussian
- We use a 5-fold CV for tuning on the inside, to optimize accuracy (ACC) and 10-fold on the outside for nested CV
- All methods are allowed a budget of 100 evaluations

¹<https://archive.ics.uci.edu/ml/datasets/spambase>

Tuning Example

We notice here:

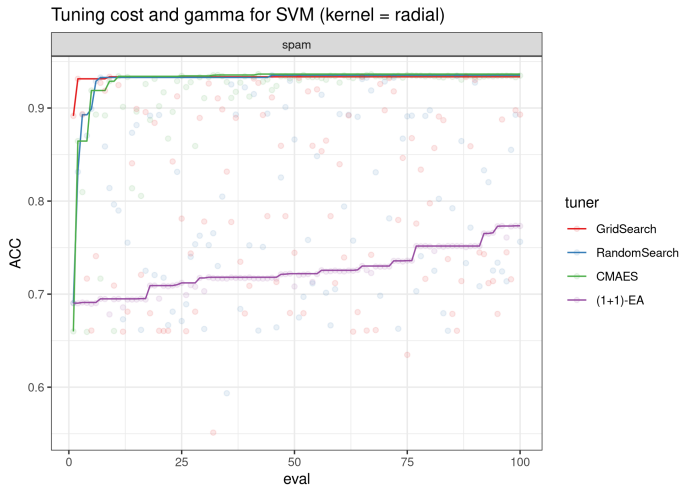
- Both *Grid search* and *random search* have many evaluations in regions with bad performance ($\gamma > 1$)
- *CMAES* only explores a small region
- *(1+1)-EA* does not converge, we probably set its control parameters in a suboptimal manner
- May we should increase ranges?
- Such a visual analysis is a lot harder for more than 2 hyperparameters



Tuning Example cont.

The *optimization trace* shows the best obtained performance until a given time point.

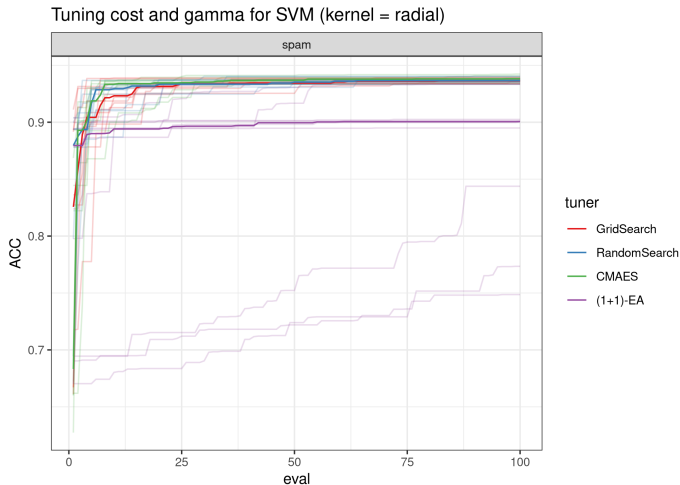
- For *random search* and *grid search* the chronological order of point evaluation is somewhat arbitrary
- The curve shows the performance on the tuning validation (*inner resampling*) on a single fold



Tuning Example cont.

The *optimization trace* shows the best obtained performance until a given time point.

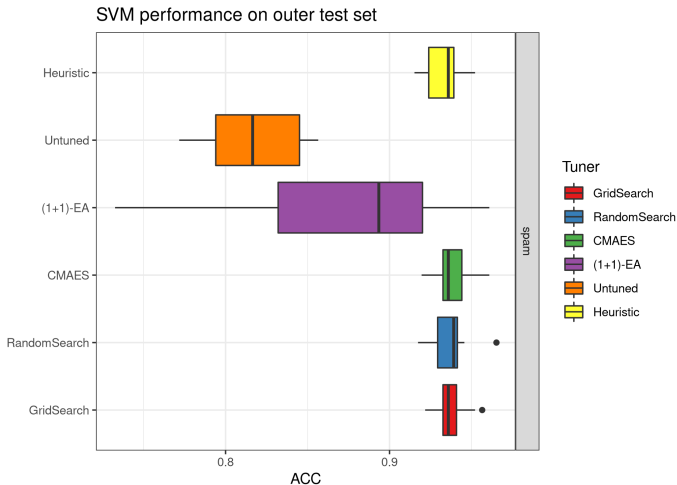
- The outer 10-fold CV gives us 10 optimization curves.
- The median at each time point gives us an estimate of the average optimization progress.



Tuning Example: Validation

Distribution of the ACC-values on *outer test sets* with a 10-fold CV.

- We compare with SVM in (unreasonable) defaults ($\text{cost} = 1, \gamma = 1$) and the previously discussed heuristic with $\text{cost} = 1$
- We abstained from proper statistical testing here
- The performance is somewhat lower than indicated by the results on the inner resampling



Challenges and Final Comments I

- ① Getting it right which hyperparameters to tune, in what ranges, and where defaults and where heuristics might be ok.
- ② Choosing and balancing out budget for tuning and inner and outer resampling.
- ③ Dealing with multi criteria-situations, where multiple performance metrics are of interest
- ④ Dealing with parallelization and time-heterogeneity
- ⑤ Ensuring the computational stability of the tuning process and dealing with crashes

Challenges and Final Comments II

- ⑥ Post-Hoc analysis of all obtained tuning results
- ⑦ Exploiting the multi-fidelity property of ML training (suppress bad configurations early without investing too much time)
- ⑧ Including preprocessing and full pipelining into the tuning process, and dealing with complex hierarchical spaces