AutoML: Hyperparameter Optimization

Example and Practical Hints

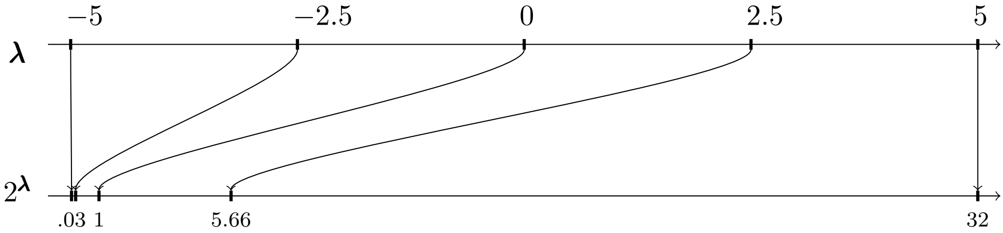
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Tuning Ranges I

- Knowledge about hyperparameters can help to guide the optimization
- For example, it can be beneficial to sample hyperparameters on a non-uniform scale.

Example: regularization hyperparameter (C or *cost*) of SVM: $[2^{-5}, 2^{5}] \approx [0.03, 32]$

- The distance between 10 and 20 should be the same as between 0.1 and 0.2.
- We might want to sample here from from a log-scale, e.g., $[2^{\boldsymbol{\lambda}_l}, 2^{\boldsymbol{\lambda}_u}]$ with $\boldsymbol{\lambda}_l = -5$ and $\boldsymbol{\lambda}_u = 5$.



Tuning Ranges II

- Similar to the scale, e.g., linear or logarithmic, the ranges in which to search have to be specified.
- Deep knowledge about the inner working of the machine learning algorithm can help to choose better bounds.
- If $\hat{\boldsymbol{\lambda}}$ is close to the border of $\boldsymbol{\Lambda}$ the ranges should be increased (or a different scale should be selected).
- Meta-Learning can help to decide which hyperparameters should be tuned in which ranges.

Tuning Example - Setup

We want to train a *spam detector* on the popular Spam dataset[1].

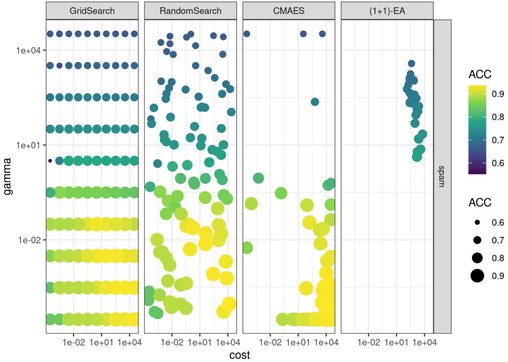
- The learning algorithm is a support vector machine (SVM) with RBF kernel.
- The hyperparameters we optimize are
  - Cost parameter cost $\in [2^{-15}, 2^{15}]$.
  - Kernel parameter $\gamma \in [2^{-15}, 2^{15}]$.
- We compare four different optimizer
  - Random search
  - Grid search
  - A $(\mu + \lambda)$-selection evolutionary algorithm with $\mu = 1$, $\lambda = 1$ and Gauss mutation with $\sigma = 1$.
  - *CMAES* - an evolutionary algorithm that generates offspring from a multivariate normal distribution (We didn't cover this algorithm).
- We use a 5-fold cross-validation and optimize the accuracy (ACC).
- 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.

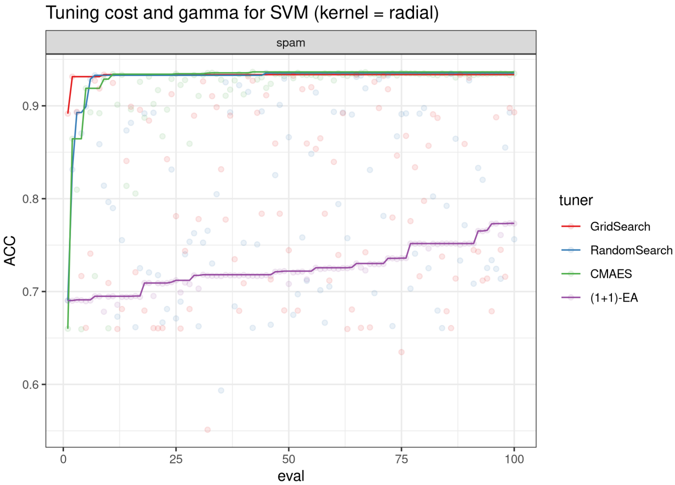


Tuning Example cont.

The *optimization curve* shows the best found configuration until a given time point.

Note:

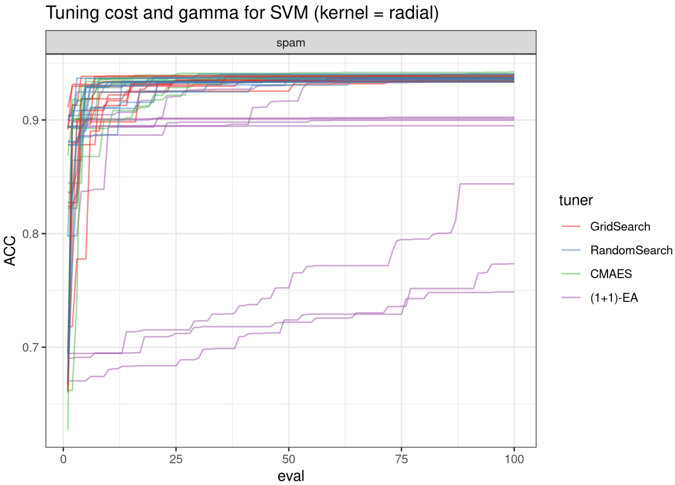
- For *random search* and *grid search* the chronological order on the x-axis is arbitrary.
- The curve shows the performance on the tuning validation (*inner resampling*) on a single fold



Tuning Example cont.

The *optimization curve* shows the best found configuration until a given time point.

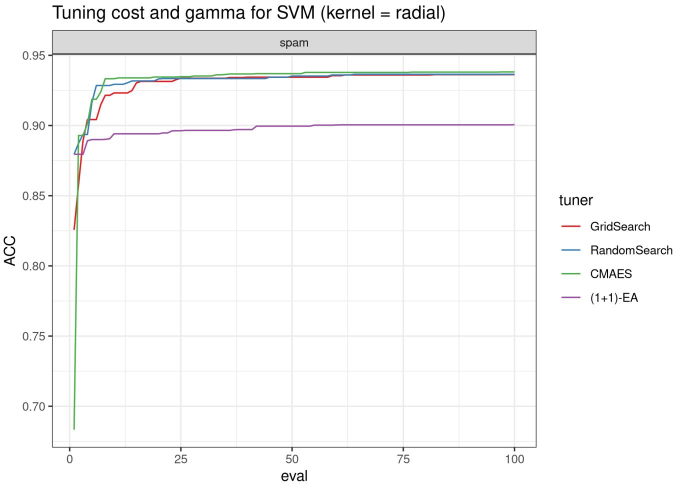
- The outer 10-fold CV gives us 10 optimization curves.



Tuning Example cont.

The *optimization curve* shows the best found configuration until a given time point.

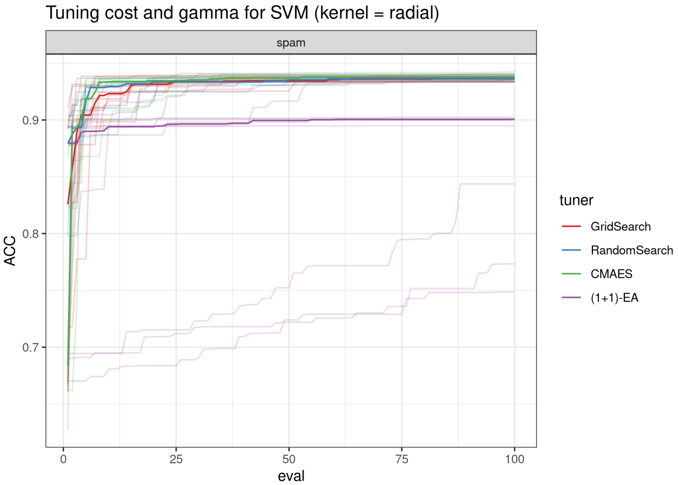
- The outer 10-fold CV gives us 10 optimization curves.
- The median at each time point gives us an estimate of the average optimization progress.



Tuning Example cont.

The *optimization curve* shows the best found configuration until a given time point.

- Remember: The final model will be trained on the *outer training set* with the configuration $\hat{\lambda}$ that lead to the best performance on the *inner test set*.
- To compare the effectiveness of the tuning strategies we have to look at the performance that $\hat{\lambda}$ gives us on the *outer test set*.

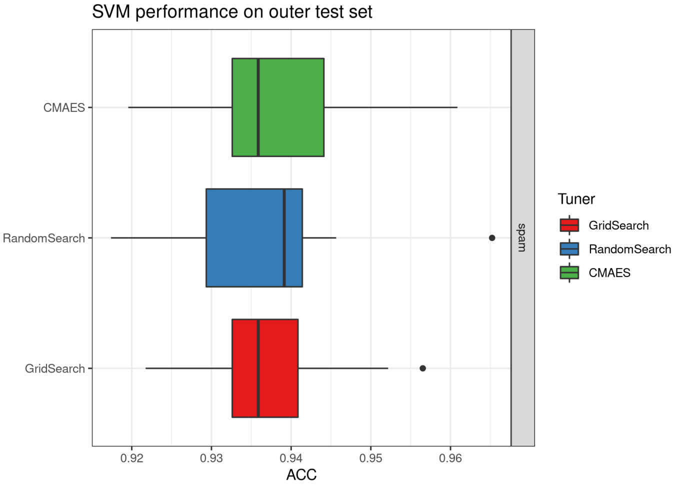


Tuning Example: Validation

The box plots show the distribution of the ACC-values that were measured on the *outer test set* with a 10-fold CV.

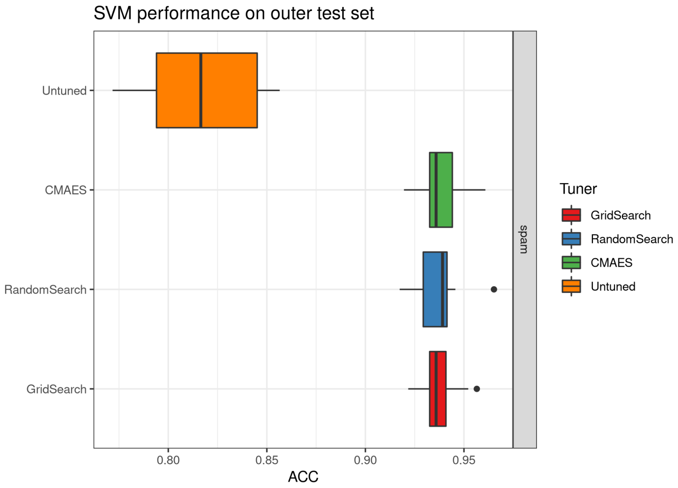
Note:

- The box plots do not indicate significant differences.
- The performance differs from the results obtained on the inner resampling.



Tuning Example: Validation

- Comparison with an SVM that was configured with $\boldsymbol{\lambda} = (\text{cost}, \gamma) = (1, 1)$ shows that tuning is necessary.

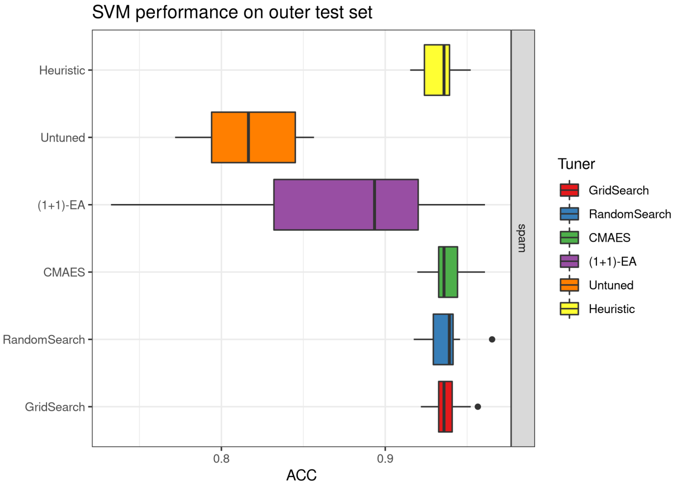


Data Dependent Defaults

- Static defaults of hyperparameters, e.g., $\boldsymbol{\lambda} = (\text{cost}, \gamma) = (1, 1)$ are rarely a good choice.
- A simple extension is to compute defaults based on some simple dataset characteristics.
- The best know example is the formula for the size of the random subset of features to consider as a split in a random forest: $\sqrt{p}$, where p is the number of features.
- For the RBF-SVM a data dependent default for the γ parameter can be computed by
  - The pairwise distances $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ between points of a random subset containing 50% of the data points are calculated.
  - The estimate is based upon the 0.1 and 0.9 quantile of these distances.
  - Basically any value between those two bounds will produce good results.
  - Take the mean of the 0.1 and 0.9 quantile of these distances as an estimate for γ.
- These simple defaults can work well and don't require expensive tuning procedures.

Tuning Example: Validation

- With the previously discussed data depended default of γ and cost $= 1$.


SVM performance on outer test set
Heuristic
Untuned
(1+1)-EA
CMAES
RandomSearch
GridSearch
ACC
spam
Tuner
GridSearch
RandomSearch
CMAES
(1+1)-EA
Untuned
Heuristic