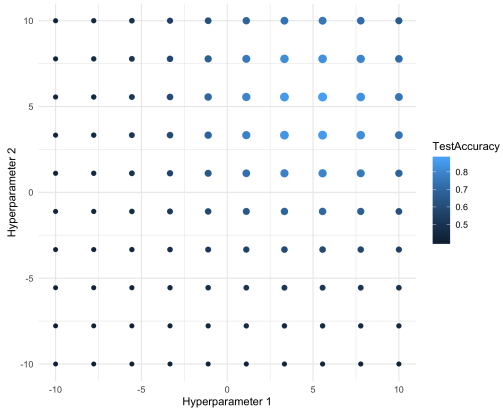
AutoML: Hyperparameter Optimization

Grid and Random Search

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Grid search I

- Simple technique which is still quite popular, tries all HP combinations on a multi-dimensional discretized grid
- For each hyperparameter a finite set of candidates is predefined
- Then, we simply search all possible combinations in arbitrary order



Grid search over 10x10 points

Grid search II

Advantages

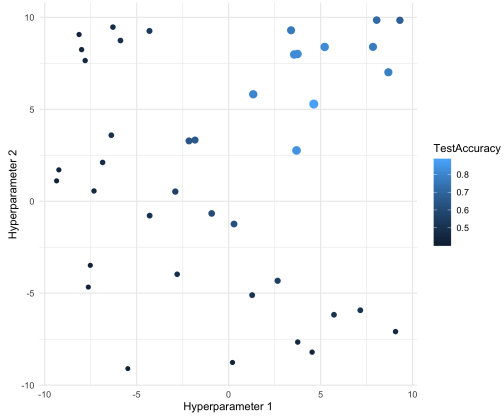
- Very easy to implement
- All parameter types possible
- Parallelizing computation is trivial

Disadvantages

- Scales badly: Combinatorial explosion
- Inefficient: Searches large irrelevant areas
- Low resolution in each dimension
- Arbitrary: Which values / discretization?

Random search I

- Small variation of grid search
- Uniformly sample from the region-of-interest



Random search over 100 points

Random search II

Advantages

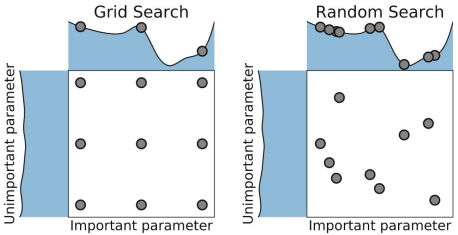
- Like grid search: Very easy to implement, all parameter types possible, trivial parallelization
- Anytime algorithm: Can stop the search whenever our budget for computation is exhausted, or continue until we reach our performance goal.
- No discretization: each individual parameter is tried with a different value every time

Disadvantages

- Inefficient: many evaluations in areas with low likelihood for improvement
- Scales badly: high dimensional hyperparameter spaces need *lots* of samples to cover.

Grid search vs. Random search

- With a tuning budget of T only $T^{\frac{1}{d}}$ unique hyperparmeter values are explored for each $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_d$ in a grid search.
- Random search will (most likely) see T different values for each hyperparameter.
- Grid search can be disadvantageous if some hyperparameters have little or no influence on the model.



Comparison of grid search and random search. [Hutter et al. 2019]