# Hyperopt

1. Docs & Github (see links in slides)
2. Search algorithms

* Random Search: rand.suggest
* Annealing -> a SMBO with GP alternative: anneal.suggest
* Tree-structured Parzen Estimators (TPE): tpe.suggest
* The search algorithm is selected within the minimization driver – fmin

Search = fmin(

fn = objective,

space = param\_grid,

max\_evals = 50,

rstate = np.random.default\_rng(42),

algo=tpe.suggest,

# algo = anneal.suggest,

# algo = rand.suggest,

)

1. Objective function

* Objective function created by user
* Pros:
* Fully customizable
* Works with any algorithm / package
* Cons:
* Needs to be coded

1. Hyperopt-Sklearn

* Alternative library to work with scikit-learn classes
* Link: see slides
* Thin documentation
* Not widely adopted

1. Search space

* Built-in module to create hyperparam sample spaces – hp
* Samples Reals, Integers, and Categories
* Extremely versatile, allows multiple distributions and combinations
* Accepts lists, tuples, and dictionaries
* Accepts nested hyperparam spaces
* Search space configuration
* Lists, tuples, and dictionaries:

from Hyperopt import hp

list\_space = [

hp.uniform(‘a’, 0, 1),

hp.loguniform(‘b’, 0, 1)

]

tuple\_space = (

hp.uniform(‘a’, 0, 1),

hp.loguniform(‘b’, 0, 1)

)

dict\_space = {

‘a’: hp.uniform(‘a’, 0, 1),

‘b’: hp.loguniform(‘b’, 0, 1)}

* Nested spaces

Nested\_space = [

[

{‘case’: 1,

‘a’: hp.uniform(‘a’, 0, 1)},

{‘case’: 2,

‘b’: hp.loguniform(‘b’, 0, 1)}

],

‘extra literal string’,

hp.randint(‘r’, 10)

]

1. Acquisition function

* Built-in acquisition functions – Expected Improvement (EI)
* EI evaluated at binomial distributions of the input space for discrete and categorical hyperparams
* EI evaluated with CMA-ES for the continuous hyperparam space

1. Analysis: Trials

* Trials object allows to store as much info as you like
* Built-in functions for plotting
* Not very useful
* No documentation ☹

1. Parallelization – MongoDB

* Allows search in parallel utilizing MongoDB
* Reduced efficiency per (single) evaluation
* Increased overall efficiency by saving time
* See code snippet in slides