Hyperparameter Optimization with hyperopt

Tree-Structured Parzen Estimation: An Expected Improvement algorithm

Based on [Algorithms for Hyper-Parameter Optimization (https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf)](https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf) from Bergstra, *et. al.*, published in NIPS 2011 Proceedings

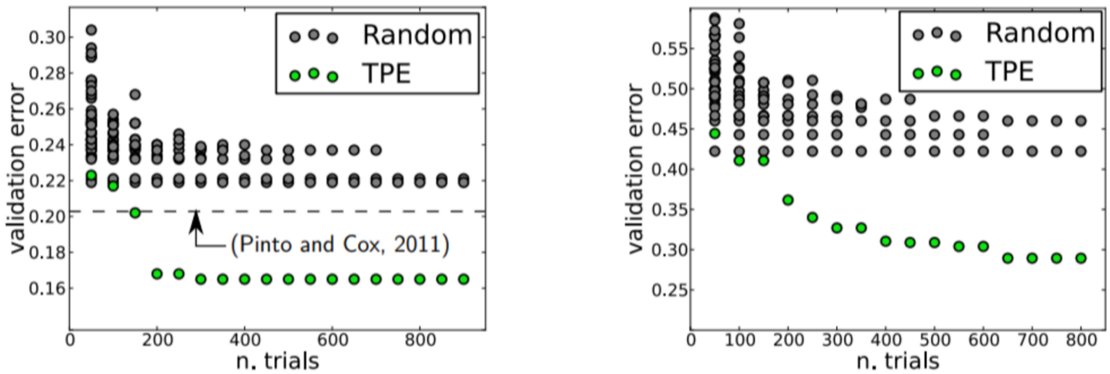
This [blog (https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f)](https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f) is great!

Machine Learning Algorithms Have Hyperparameters

Machine Learning Algorithms Have Hyperparameters

Which can leave you with hundreds of thousands to millions of combinations to search through...

- `hyperopt` uses Tree-Structured Parzen Estimation (TPE) to Model the hyperparameter search space
- As it tests new hyperparameter combinations, it updates what it knows about the search space to make increasingly informed choices



Validation Errors comparing random search and a model based approach on LFW (left) and PubFig83 (right)

[Library-Specific Implementations (https://github.com/hyperopt)](https://github.com/hyperopt) * I have not tested these, except for hyperas:

- `hyperopt` is easy to implement
- You simply provide it with
  - the **algorithm** being tested
  - **objective function** to minimize
  - the **parameters** to search
  - the **ranges of values** for each parameter
  - and the **initial distribution** of those ranges
- The optimization algorithm updates those distributions as it runs to find the highest performing areas

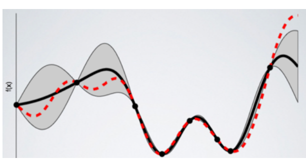
In [ ]:
```
# hyperopt is easy to implement!

from hyperopt import fmin, tpe, hp, STATUS_OK, Trials

def objective(params):
    """ algorithm with some loss function here """
    return {'loss': -acc, 'status': STATUS_OK, 'model': model}

space = {
        'epsilon'  : hp.choice('base_epsilon',
                               [10**1,10**0,10**-1,10**-2,10**-3,10**-4]),
        'momentum' : hp.quniform('initial_momentum', 0.0,0.9,0.1)
        }

best = fmin(objective,
            space=space,
            algo=tpe.suggest,
            max_evals=100,
            trials=Trials())
print(best)
```

Tree-Structured Parzen Estimators : Sequential Model-Based Optimization (SMBO)

- **Domain**: Search Space of all possible Hyperparameter combinations, represented in a Tree-Structure
-

Objective Function	Selection Criteria
$\mathcal{L}(y - \hat{y}) = -\sum_{i=1}^{n} y_i \log \hat{y}_i$	$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y \vert x) dy$
$x^* = \arg\min_{x \in \mathcal{X}} f(x)$	$\gamma = p(y < y^*)$ $p(x \vert y) = l(x)$ if $y < y^*$ $= g(x)$ if $y \geq y^*$

- **Surrogate Model**: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$
- **History**: Keep track of the past and become increasingly "less wrong" with hyperparameter choices



Thanks!