Tree Parzen Estimators With Uncertainty For Hyperparameter Optimization Of Machine Learning Algorithms

Alejandro Morales-Hernández [1,2,*], Inneke Van Nieuwenhuyse [1,2], Sebastian Rojas Gonzalez [1,2,3]

[1] VCCM Core Lab, Flanders Make, Belgium
[2] Data Science Institute, Hasselt University, Belgium
[3] Surrogate Modelling Lab, Ghent University, Belgium

This research is supported by the Flanders Artificial Intelligence Research Program

*Corresponding author:
alejandro.moraleshernandez@uhasselt.be

1

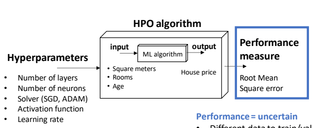
Outline

Keywords:
- Hyperparameter optimization
- Bayesian optimization
- Performance uncertainty

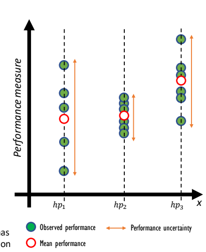
2

Hyperparameters:

- Influence the learning process
- Are not optimized during the training of the ML algorithm
  - should be specified before the training phase
- Complex domain (numeric, discrete, etc)
  - Hyperparameter optimization (HPO) = hard!



Performance = uncertain
- Different data to train/validate (bootstrapping, k-fold cv, …)
- Retraining if the ML algorithm has some (random) inner optimization
- Monte Carlo dropout (for DNN)



3

Hyperparameter optimization problem

Validation protocol

Performance measure

Validation dataset

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\, V(f \mid A_\lambda, D_{train}, D_{validation})$$

Optimal hyperparameter configuration

Training dataset

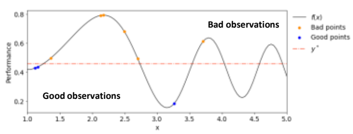
The algorithm A with its hyperparameters instantiated to a configuration λ is denoted by A_λ

Goal of this research:
- **Data efficient search** for an optimal configuration
- Account for **performance uncertainty** during the optimization

4

Tree Parzen Estimators (TPE) algorithm*

Model the distribution of each input using KDE



y^* is some quantile γ of the observed $\boldsymbol{Y}$ values, so that $p(y < y^*) = \gamma$

$$P(x_*|X,Y) = \begin{cases} l(x) & y < y^*, x \in \boldsymbol{X} \\ g(x) & other\ case \end{cases}$$

How to suggest a new hyperparameter configuration?

ESTIMATE DENSITIES $l(x)$ and $g(x)$ using all the observations and their probability of being "good" and "bad"

SAMPLE n_c candidates from $l(x)$

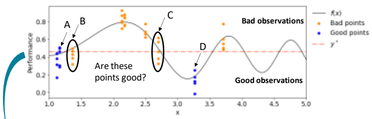
SELECT x^* that maximizes the ratio $\frac{l(x)}{g(x)}$

* Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems, 24*

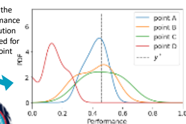
5

Tree Parzen Estimators (TPE) with uncertainty (our algorithm)

Model the distribution of each input using KDE



KDE of the performance distribution observed for each point

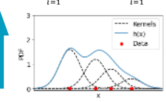


Each point has a probability of being good, and a probability of being bad

- WEIGHT the importance of the points in estimating l(x) and g(x) using KDE in input space

How important is each point in estimating the distribution?

$$h(x) = \frac{1}{Nh}\sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \sum_{i=1}^{N} w_i = 1$$

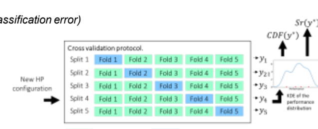


6

Experimental design

Binary classification problems (minimization of classification error)

HP Multilayer Perceptron (d=5)		
max_iter	$\mathbb{Z}$	$U[10, 1000]$
neurons	$\mathbb{Z}$	$U[5, 1000]$
lr_init	$\mathbb{Z}$	$-\mathrm{U}\ \log([10^{-6}, 10^{-1}])$
b1	$\mathbb{R}$	$\log([10^{-7}, 1])$
b2	$\mathbb{R}$	$\log([10^{-7}, 1])$



Classification problems

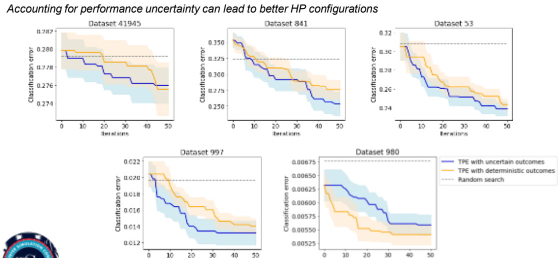
Dataset	OpenML ID	Features	Instances
Balance scale	997	4	625
Optdigits	980	64	5620
Stock	841	9	950
Heart-statlog	53	13	270
Ilpd	41945	10	583

Algorithms setting	
Initial design space	$11d - 1 = 54$
Replications to account for uncertainty (k in cross-validation protocol)	5
Iterations	50
Candidates to sample per iteration (n_c)	2000
TPE(γ)	0.2
Macro-replications	10

7

Results

Accounting for performance uncertainty can lead to better HP configurations



8

Concluding remarks

- Basic idea: TPE algorithm adjustment to account for performance uncertainty
  - Weighted KDE: probability that a given HP configuration is "good" or "bad"
- Our proposal outperforms the original TPE (final result and/or search speed)
  - Interesting for settings with limited budget!
- Further fine-tuning required to get high-quality performance on datasets with probability of being good and bad very close (dataset 980)

Further research:

- Multi-objective extension
- Multivariate KDE

9

Thanks

Q & A

10