Enhancing LA-MCTS with SMAC for Heterogeneous Search Space

Nov 29th, 2023

Wenxuan Li

Background: LA-MCTS (W. Lin et al. 2020)

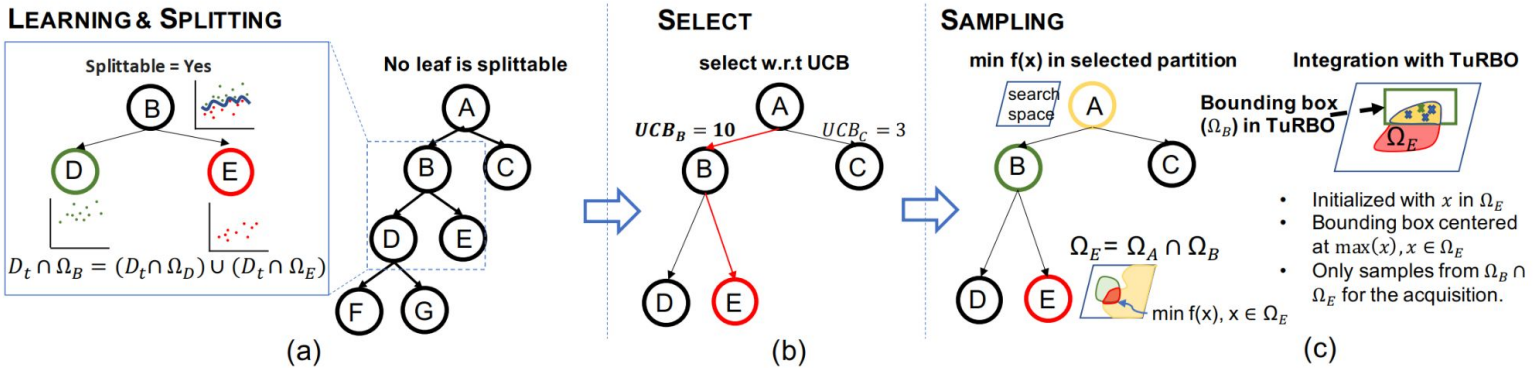


Figure 2: **The workflow of LA-MCTS**: In an iteration, LA-MCTS starts with building the tree via splitting, then it selects a region based on UCB. Finally, on the selected region, it samples by BO.

Background: Heterogeneous Search Space

LA-MCTS has integrated GP-based BO algorithms as its local samplers

- Basic form of GP-BO assumes that the inputs are continuous
- But there are:
  - Discrete/Ordinal Variables (e.g. number of layers in an MLP)
  - Categorical Variables (e.g. choice of activation functions for a MLP layer)

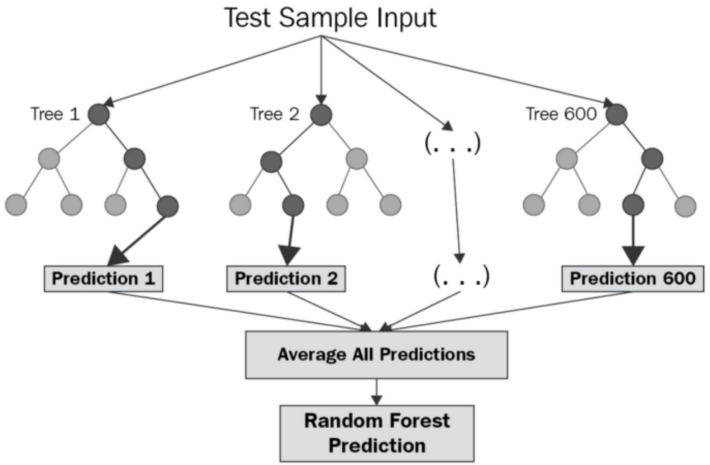
GP-BO does not inherently support them and needs further pre-/post- processing techniques

Approach: SMAC (F. Hutter et al. 2011)

Sequential Model-based Algorithm Configuration (SMAC)

Algorithm 1 Bayesian Optimization Framework

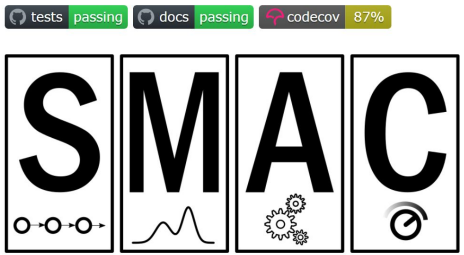
1: **Input:** Objective function f, Initial data D_0, Acquisition function a, Model M
2: Initialize dataset $D \leftarrow D_0$
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Fit model M to D
5: Find x_t by optimizing acquisition function: $x_t = \arg\max_x a(x|D, M)$
6: Evaluate objective function: $y_t = f(x_t)$
7: Augment data: $D \leftarrow D \cup \{(x_t, y_t)\}$
8: **end for**
9: **Output:** Best observed x and corresponding y



Random Forest

Approach: SMAC3 (M. Lindauer et al. 2021)

SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization



SMAC offers a robust and flexible framework for Bayesian Optimization to support users in determining well-performing hyperparameter configurations for their (Machine Learning) algorithms, datasets and applications at hand. The main core consists of Bayesian Optimization in combination with an aggressive racing mechanism to efficiently decide which of two configurations performs better.

SMAC3 is written in Python3 and continuously tested with Python 3.8, 9.9, and 3.10. Its Random Forest is written in C++. In further texts, SMAC is representatively mentioned for SMAC3.

> Documentation

> Roadmap

```python
from ConfigSpace import Configuration, ConfigurationSpace

import numpy as np
from smac import HyperparameterOptimizationFacade, Scenario
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score

iris = datasets.load_iris()


def train(config: Configuration, seed: int = 0) -> float:
    classifier = SVC(C=config["C"], random_state=seed)
    scores = cross_val_score(classifier, iris.data, iris.target, cv=5)
    return 1 - np.mean(scores)


configspace = ConfigurationSpace({"C": (0.100, 1000.0)})

# Scenario object specifying the optimization environment
scenario = Scenario(configspace, deterministic=True, n_trials=200)

# Use SMAC to find the best configuration/hyperparameters
smac = HyperparameterOptimizationFacade(scenario, train)
incumbent = smac.optimize()
```

Evaluation

Principle: High-dimensional and Heterogeneous Search Space

- Current thinking: ML model hyperparameter optimization tasks

Reference

- F. Hutter et al., Sequential Model-Based Optimization for General Algorithm Configuration. 2011.
- L. Wang et al. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. 2020
- M. Lindauer et al. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. 2021

Thank you