Replacing The Acquisition Function in Bayesian Optimization by a Neural Network

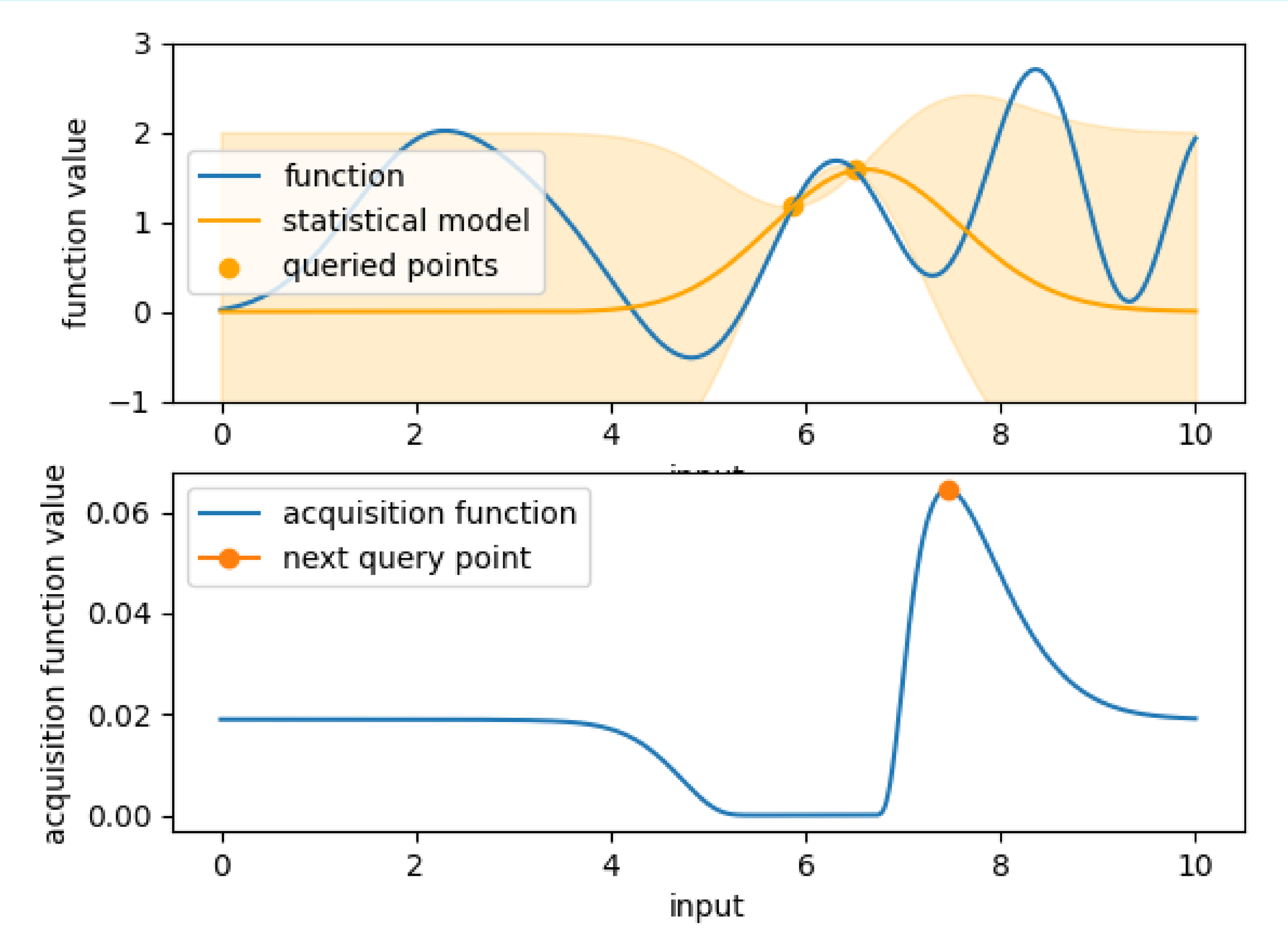
Author: Shayan Ramezani

Supervisors: Matthijs Spaan, Joery de Vries

1. Introduction

General Problem: Optimizing control variates of complex systems without (strong) prior knowledge of the underlying dynamics and with limits involved. These systems are called objective functions here.

Current State of Research: A lot of progress made with Bayesian Optimization (BO) but suboptimal performance for specialized tasks.



Bayesian optimizaiton example

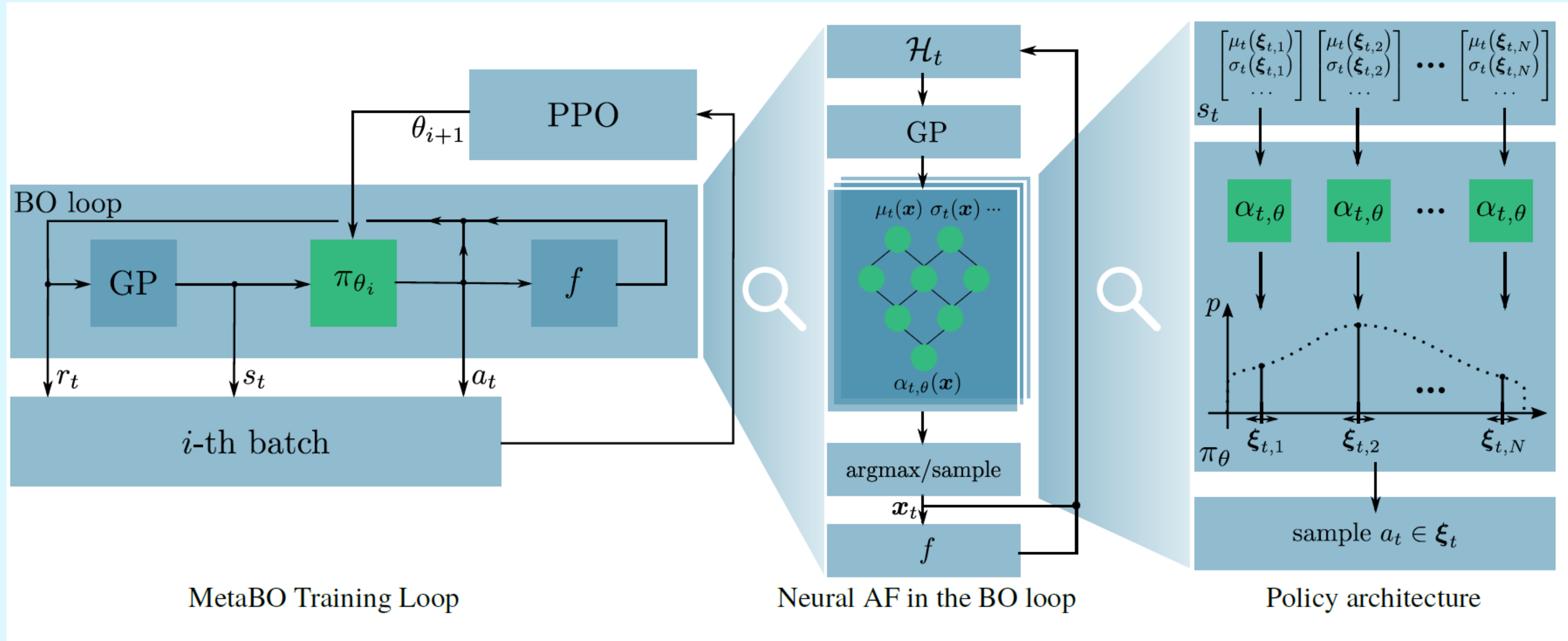
Solution Offered: Transfer learning between related tasks with help of neural networks.

Research Question: How effectively do meta-learned acquisition functions in Bayesian optimization perform when optimizing for control variates of unknown functions, as compared to BO with standard acquisition functions?

Contributions of the paper: Answer the research question by using the *BBox library* to generate objective functions and by conducting tests on these functions..

2. Implementation

1. **Main change in BO:** Replace the standard acquisition function (AF) with a neural network (NN) AF. Call it the MetaBO algorithm.



High-level overview of the algorithm [1]

2. **Agent:**
- Inspired by the actor-critic network:
  - **actor** (NN + action selector):
    - NN input: the environment state, which is the composite of actions, corresponding means and standard deviations from GP, step, and budget.
    - NN output: value for each inputted action.
    - Action selector selects the next action by building a distribution around the inputs/outputs.
  - **critic (just an NN):**
    - Predict the cumulative reward from knowing current step and budget.

3. **Environment:**
- Encapsulates the objective function and estimates its optimum for reward computation as negative simple regret.
- Contains statistical model (GP) as part of its state.
- To input this state to the actor, discretization is needed:
  i. First discretize the full state;
  ii. Evaluate the actor's NN for this set;
  iii. Discretize the state around the top k points from previous step;
  iv. The set from step i and iii forms a discretized portray of the state.

4. **Training Loop has two NNs to train**:
- **Actor**: aims to select the best point to evaluate next:
  a. Input the discretized state from 3-iv to get the next point to evaluate the objective function for;
  b. Get the reward for this next point;
  c. After repeating this for some time, update the actor based on the collected actions+rewards.

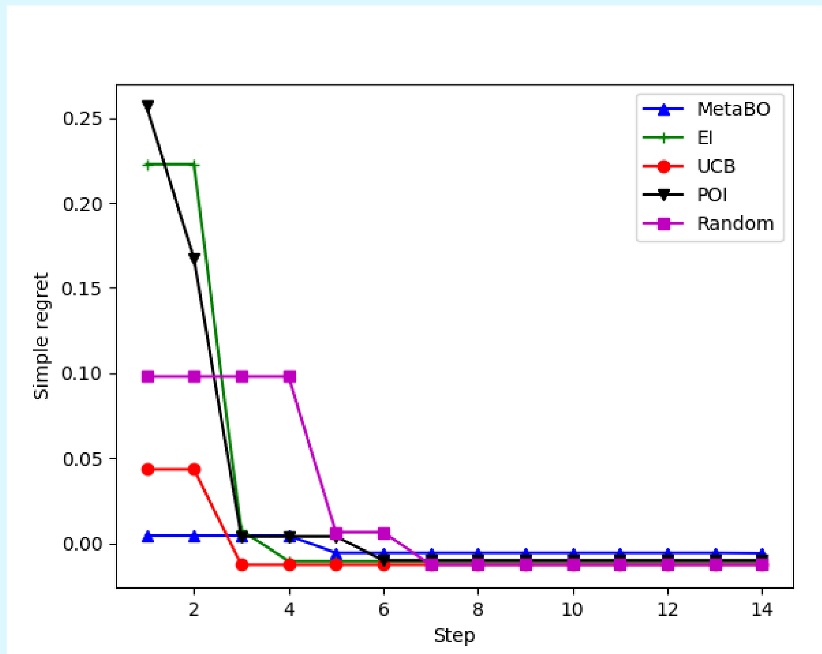
- **Critic**: aims to learn a value function to predict the expected cumulative reward from a state

3. Evaluation

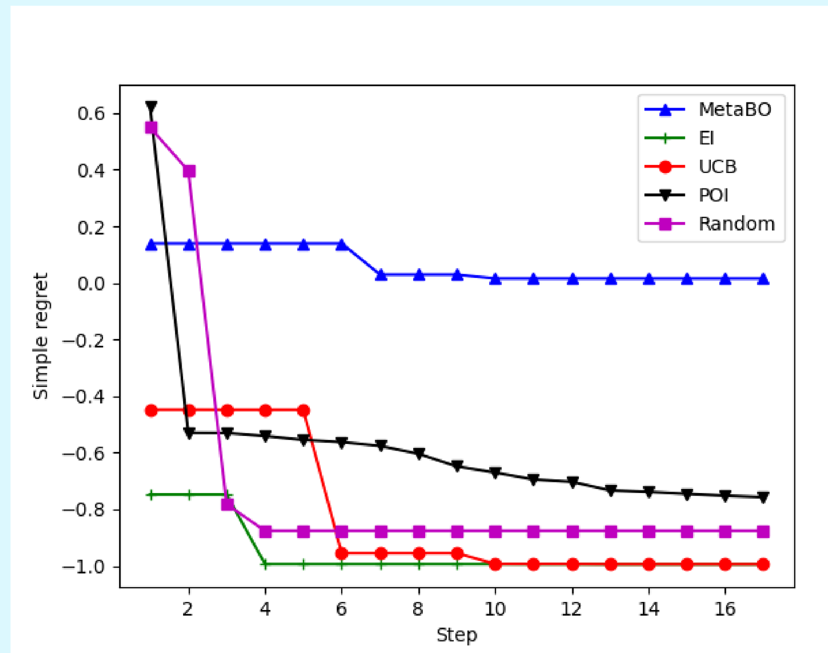
- Evaluation conducted on two different group of functions: Simple Convex function & Gaussian process functions

Results

- Altogether, the MetaBO algorithm does not surpass the performance of standard BO.
- This is mainly due to the agent not learning as the losses during training do not converge



Simple convex function evaluation



Gaussian process function evaluation

4. Conclusions

- Subpar performance of the algorithm has been observed
- Partially due to lack of experience of the author with Reinforcement learning
- This also underscores the complexity involved in implementing the algorithm as there numerous plug-and-play reinforcement learning algorithms that did not perform well here.

5. Future works

- Hope with this work is to give more clarity on steps involved to implement the MetaBO algorithm and testing with the *BBox library*.
- Improvements in later work can be the modelling of objective functions and modifying the input of the critic network.
- Application of transfer learning to multiple components of BO concurrently.

References

[1] M. Volpp et al., Meta-learning acquisition functions for transfer learning in Bayesian optimization, 2020. arXiv: 1904.02642 [stat.ML].

TU Delft