Enhancing Active Learning in Emukit: an extended experimental analysis
Emukit: Emulation of physical processes with Emukit

- A broad range of contracts for: Bayesian Optimisation, experimental design, sensitivity analysis, quadrature, and multi-fidelity emulation
- Model-agnostic backend
- Related active learning frameworks:
  - Focused on single tasks (e.g., BayesOpt)
  - Tightly-coupled with a modelling framework
- Case studies (e.g., BO for quantum computer memory)
import numpy as np

class IModel:
    def predict(self, X: np.ndarray) -> Tuple[np.ndarray, np.ndarray]:
        """
        Predict mean and variance values for given points
        """
        raise NotImplementedError

    def set_data(self, X: np.ndarray, Y: np.ndarray) -> None:
        """
        Sets training data in model
        """
        raise NotImplementedError

Reference: github.com/EmuKit/emukit
Proposed Open-Source Contributions
1. On Bayesian Optimisation

- Emukit supports a deluge of acquisition functions

Proposal
- Thompson Sampling acquisition function
- New contract for the surrogate Model
- Thompson Sampling requires a function sample from the prior

<table>
<thead>
<tr>
<th>Library</th>
<th>Built-in model</th>
<th>Acquisition function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiceOptim</td>
<td>GP</td>
<td>EI, EQI, KG</td>
</tr>
<tr>
<td>laGP</td>
<td>GP</td>
<td>EI</td>
</tr>
<tr>
<td>mlrMBO</td>
<td>GP, RF</td>
<td>EI, EQI, UCB, PM</td>
</tr>
<tr>
<td>Spearmint</td>
<td>GP</td>
<td>EI, predictive ES</td>
</tr>
<tr>
<td>GPyOpt</td>
<td>GP, RF</td>
<td>EI, ES, UCB, PoI</td>
</tr>
<tr>
<td>Cornell-MOE</td>
<td>GP</td>
<td>EI, KG, predictive ES</td>
</tr>
<tr>
<td>GPflowOpt</td>
<td>GP</td>
<td>EI, UCB, PoF, PoI, max-value ES</td>
</tr>
<tr>
<td>pyGPGO</td>
<td>GP, GBM, RF, tSP</td>
<td>EI, PoI, UCB, predictive ES</td>
</tr>
<tr>
<td>Emukit</td>
<td>GP</td>
<td>EI, ES, UCB, PoF, PoI, max-value ES</td>
</tr>
<tr>
<td>Dragonfly</td>
<td>GP</td>
<td>EI, PoI, TS, UCB</td>
</tr>
<tr>
<td>Trieste</td>
<td>GP</td>
<td>EI, UCB, PM, PoF, TS, max-value ES</td>
</tr>
<tr>
<td>BoTorch</td>
<td>GP</td>
<td>EI, KG, PM, PoI, UCB, max-value ES</td>
</tr>
</tbody>
</table>

Table from Zhang, Mimi, et al. 2021
Thompson Sampling Acquisition

Full paper citation: Bingham, Eli, et al. 2019
TS-BO: Motivation

- Thompson Sampling Bayesian Optimization (TS-BO) can be extended to perform evaluations in parallel.

- A set of “n” sequential evaluations are equivalent to “n” (a)synchronous evaluations across “T” threads/devices.

- Proof: Kandasamy, Kirthevasan, et al. 2018

- Parallel Bayesian Optimisation with Emukit
2. On Experimental Design

Uncertainty Sampling
- Select most ‘uncertain’ data point

Integrated Variance Reduction
- Expected Error Reduction
- Evaluate ‘x’ if it minimises the future variance
Density Weighting

- High-variance samples may be isolated
- Sample from populated regions
- \( dw(x) = us(x) \times \text{density}(x) \)

Solutions
- Kernel density
- Approximations (Settles, Burr, 2009)

Example decision boundary for classification task.
Bayesian Experimental Design
Research Contributions
An extended analysis of Emukit’s active learning

- An extended analysis with respect to existing methods in Emukit
- Comparison with BoTorch on Bayesian Optimisation

Extensions
- Hyper-parameter tuning case study
- Parallel TS - Bayesian Optimisation

Future vision
- Original proposal: Multi-objective Bayesian Optimisation (MOBO)
Investigate Emukit’s open source codebase
Devise a plan for practical contributions
Design a research methodology
Engineer the proposed contributions in Emukit
Evaluate the methods based on the methodology
Case study extension
Pull request
Thank you!
Appendix: Emukit Workflow

1. while stop condition is False:
   2. acquire sample \textquoteleft x\textquoteright{} based on emulator
   3. run experiment with sample \textquoteleft x\textquoteright{}
   4. update the emulator with the observed behaviour

User defines the business problem and injects the model into the Active Learning loop
References
