BOAT
Building Auto-Tuners with Structured Bayesian Optimization

Authors: Valentin Dalibard, Michael Schaarschmidt, Eiko Yoneki
Motivation

The complexity of modern Machine Learning systems has led to a sharp increase in the number and sensitivity of hyper-parameters necessary to tune them.

Problems:
- The curse of dimensionality
- Training time limits fitness evaluations
- Highly distributed
# Classical Bayesian Optimization

<table>
<thead>
<tr>
<th>Non-parametric Model</th>
<th>Acquisition Function</th>
<th>Gaussian Process</th>
</tr>
</thead>
</table>
| - Grows in complexity based on the data  
- Can model any function given enough samples | - Encodes the exploration-exploitation trade-off  
- It is not guaranteed to converge, especially in high-dimensional spaces | - Common non-parametric model  
- Entirely defined by its mean and covariance function |
Bayesian Optimization

**Input:** Objective function $f()$

**Input:** Acquisition function $\alpha()$

1: Initialize the Gaussian process $G$
2: for $i = 1, 2, \ldots$ do
3:   Sample point: $x_t \leftarrow \arg\max_x \alpha(G(x))$
4:   Evaluate new point: $y_t \leftarrow f(x_t)$
5:   Update the Gaussian process: $G \leftarrow G \mid (x_t, y_t)$
6: end for
Bayesian Optimization is not guaranteed to converge in high-dimensional (>10) domains

Reasons:

- The curse of dimensionality
  - Tackled by SBO
- Non-convergence of the acquisition function
  - Requires complex decomposition and algorithms
Structured Bayesian Optimization

- **Probabilistic Programming**
  - Draw values from random distributions
  - Constrain variable values to those observed
  - Output variable distribution

- **SBO:**
  - Structured Bayesian Optimization allows for the injection of domain knowledge into Bayesian Optimization
  - In the form of a probabilistic program

```
# Draw from distributions
bias = uniform_draw(0.0, 1.0)
flip = bernoulli_draw(bias)

# Observe an outcome
observe(flip, true)

# Output the resulting distribution
predict(bias)
```

**Listing 4.1:** A very simple probabilistic program.

![Figure 1: Procedure of Structured Bayesian Optimization](image-url)
Semi-Parametric Models

- Semi-Parametric Model:
  - Custom parametric model is encoded in probabilistic program
  - Non-parametric program learns the difference between the actual data and the parametric program

```cpp
double predict(int ygs, int sr, int mtt) {
    return gp.predict({ygs, sr, mtt}) + parametric(ygs, sr);
}

double observe(int ygs, int sr, int mtt, double observed_rate) {
    return gp.observe({ygs, sr, mtt},
                      observed_rate - parametric(ygs, sr));
}
```

(a) Parametric (Linear regression)  (b) Non-parametric (Gaussian process)  (c) Semi-parametric (Combination)
Boat is meant to allow for the easy construction of a bespoke auto-tuner by defining semi-parametric models of the system.

It requires defining:

- The configuration space
- Objective function and metrics
- Probabilistic Program to model system behaviour
Efficiency

In order to make high-dimensional optimization problems tractable, BOAT has several restrictions:

- The larger model must be split into multiple components.
- Components should only predict one value.
- Components should be assembled into a model assuming conditional independence.
Garbage Collection Case Study

Figure 2: Dataflow of our garbage collection model
Garbage Collection Case Study

Figure 2: Dataflow of our garbage collection model

```c
struct CassandraModel : public DAGModel<CassandraModel> {
    void model(int ygs, int sr, int mtt) {
        // Calculate the size of the heap regions
        double es = ygs * sr / (sr + 2.0); // Eden space's size
        double ss = ygs / (sr + 2.0); // Survivor space's size
        // Define the dataflow between semi-parametric models
        double rate = output("rate", rate_model, es);
        double duration = output("duration", duration_model, es, ss, mtt);
        double latency = output("latency", latency_model, rate, duration, es, ss, mtt);
    }
    ProbEngine<GCRateModel> rate_model;
    ProbEngine<GCDurationModel> duration_model;
    ProbEngine<LatencyModel> latency_model;
};
```

Figure 5: Results for YCSB workloads A, B and D.

Figure 6: Convergence of the frameworks on workload B.
Neural Network Case Study

Optimizes BOAT for the notoriously difficult NN scheduling problem

- Up to 32 dimensions, compared to 10
- Highly distributed
- Difficult to evaluate
- Greatly outperforms generic auto-tuners

Figure 8: Convergence of the frameworks on Setting C using SpeechNet with a $2^{16}$ batch size.
Recap

Classical Bayesian Optimization

- Non-Parametric model
- Acquisition Function
Recap

Classical Bayesian Optimization

- Non-Parametric model
- Acquisition Function

Semi-Parametric Model
- Probabilistic Programming

Structured Bayesian Optimization
Related Work

Since Publication:

- **ProBO:**
  - Probabilistic-programming-language agnostic
  - Coming up next
- **Arrow:**
  - Same idea as BOAT, applied to cloud VM architectures
- **BoTorch:**
  - Bayesian Optimization which can leverage the PyTorch to more efficiently solve the acquisition function
  - Uses probabilistic models implemented in PyTorch

Authors Future Work Ideas:

- Allow for easier modelling of “stacked” systems where each layer depends on the previous
- Allow for use in real-time systems
- Allow for more general modelling
Critique

- Opentuner can also be customised
  - A comparison against Opentuner with a similar amount of customization and time investment could have helped strengthen the evaluation
  - Could have shown that it is either easier to customize or faster for the same amount of effort
- The Neural Network example is significantly more complex than presented in the paper, it requires different algorithms for the acquisition function alongside decomposition tricks
  - Does not lower the impact of successfully optimizing a 32 dimensional problem
  - It does indicate that solving such problems is more complex than simply defining probabilistic programs
- No mention in the final paper that the probabilistic programming library cannot handle models with more than 5 parameters, strongly implied by the encouragement to slowly add structure
- My future work ideas:
  - New acquisition function, several have been proposed
  - Rebuilding the framework structure on top of BoTorch
Questions?