BOAT: Building Auto-Tuners with Structured Bayesian Optimization

Valentin Dalibard, Michael Schaarschmidt, Eiko Yoneki
Motivation

Problem

Modern systems require fine tuning a large number of hyperparameters

Existing Solutions

Bayesian optimization

Problems

Configuration space too large

Source: Dalibard et al., 2017
BOAT: BespOke Auto-Tuner

Structured Bayesian Optimization

BOAT Framework

Case Studies

Distributed scheduling of neural network computation

Garbage collection

Source: Dalibard et al., 2017
Structured Bayesian Optimization

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

Source: Dalibard et al., 2017
Probabilistic Model: Garbage Collection

**GC Configuration Space**
- young generation size
- max tenuring threshold
- survivor ratio

**Dataflow of GC Model**
- GC Flags
- GC Average Duration Model
- GC Rate Model
- Latency Model
- Predicted Latency

*Source: Dalibard et al., 2017*
Source: Dalibard et al., 2017

BOAT: BespOke Auto-Tuner

Structured Bayesian Optimization

BOAT Framework

Case Studies

Distributed scheduling of neural network computation

Garbage Collection

Distributed scheduling of neural network computation

Source: Dalibard et al., 2017
BOAT Framework

Source: Dalibard et al., 2017
Semi-Parametric Models in BOAT

Source: Dalibard et al., 2017
Probabilistic Models in BOAT

Semi-Parametric Model

```cpp
struct GCRateModel : public SemiParametricModel<GCRateModel> {
    GCRateModel() {
        allocated_mbs_per_sec =
            std::uniform_real_distribution<>(0.0, 5000.0)(generator);
        // Omitted: also sample the GP parameters
    }

double parametric(double eden_size) const {
    // Model the rate as inversely proportional to Eden's size
    return allocated_mbs_per_sec / eden_size;
}

double allocated_mbs_per_sec;
};

int main() {
    // Example: observe two measurements and make a prediction
   _probEngine<GCRateModel> eng;
    eng.observe(0.40, 1024); // Eden: 1024MB, GC rate: 0.40/sec
    eng.observe(0.25, 2048); // Eden: 2048MB, GC rate: 0.25/sec
    // Print average prediction for Eden: 1536MB
    std::cout << eng.predict(1536) << std::endl;
}
```

Source: Dalibard et al., 2017

Directed Acyclic Graph Model

```cpp
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;
};

int main() {
    CassandraModel model;
    // Observe a measurement
    std::unordered_map<std::string, double> m;
    m["rate"] = 0.40; m["duration"] = 0.15; m["latency"] = 15.1;
    int ygs = 5000, sr = 7, mtt = 2;
    model.observe(m, ygs, sr, mtt);
    // Prints distributions (mean and stdev) of rate, duration and latency with a larger young generation size (ygs)/
    std::cout << model.predict(6000, sr, mtt) << std::endl;
    // Print corresponding expected improvement of the latency
    std::cout << model.expected_improvement("latency", 15.1, 6000, sr, mtt) << std::endl;
}
```
BOAT: BespOke Auto-Tuner

Structured Bayesian Optimization

BOAT Framework

Case Studies

Distributed scheduling of neural network computation

Source: Dalibard et al., 2017
Case Study: Garbage Collection

Configuration Space

Objective Function

cassandra

Model

Source: Dalibard et al., 2017
Case Study: Garbage Collection

BOAT vs. Cassandra Default

BOAT vs. Generic Auto-Tuners

Source: Dalibard et al., 2017
Case Study: Neural Networks

Configuration Space

Objective Function

Model

Average time for SGD

Source: Dalibard et al., 2017
Case Study: Neural Networks

Source: Dalibard et al., 2017
Case Study: Neural Networks

Source: Dalibard et al., 2017
Summary

Structured Bayesian Optimization

BOAT Framework

Case Studies

Distributed scheduling of neural network computation

Garbage Collection

Source: Dalibard et al., 2017
BOAT: 6 Years Later...

Framework

About
No description, website, or topics provided.

Readme
Apache-2.0 license
51 stars
7 watching
13 forks

Releases

Paper

BOAT: Building auto-tuners with structured Bayesian optimization
Authors: Valentin Daltbard, Michael Schaarschmidt, Elko Yonke
Publication date: 2017/4/3
Book: Proceedings of the 26th International Conference on World Wide Web
Pages: 479-488
Description: Due to their complexity, modern systems expose many configuration parameters which users must tune to maximize performance. Auto-tuning has emerged as an alternative in which a black-box optimizer iteratively evaluates configurations to find efficient ones. Unfortunately, for many systems, such as distributed systems, evaluating performance takes too long and the space of configurations is too large for the optimizer to converge within a reasonable time.

Total citations: Cited by 88

Sources: GitHub, Google Scholar
Some Thoughts on the Paper

➢ Extension of neural network case study beyond system perspective: Investigate whether BOAT could be used to increase model accuracy through hyperparameter selection (e.g. in image recognition tasks)

➢ No discussion about potential limitations and problems of the approach
  ▪ What if the modularization of the overall system is not possible or the input-output relationships are unknown?
  ▪ Are there situations in which the added knowledge could have a negative impact on the performance of the system (thinking of reward shaping in RL)?
  ▪ ...

➢ Little technical depth on how BOAT maximizes the expected improvement and performs inference
Questions / Discussion
References


Image Sources


➢ https://miro.medium.com/max/1072/1*tkpDTzQKwekXbSd0L_e9Aw.png

➢ https://cds.cern.ch/record/2702355/files/step1.png

➢ Jason Ansel et al. **Opentuner: an extensible framework for program autotuning**. In Proceedings of the 23\textsuperscript{rd} international conference on Parallel architectures and compilation, pages 303-316. ACM, 2014.

➢ https://thegradient.pub/content/images/size/w1600/2019/11/kernel_cookbook-2.png

➢ https://www.studytrails.com/2021/02/10/distributed-machine-learning-2-architecture/

➢ https://iq.opengenus.org/content/images/2018/05/jvm_memory.png

➢ https://upload.wikimedia.org/wikipedia/commons/thumb/5/5e/Cassandra_logo.svg/1200px-Cassandra_logo.svg.png

➢ https://ozzieliu.com/assets/img/gradientdescent.png

➢ https://github.com/VDalibard/BOAT

➢ https://scholar.google.com/citations?view_op=view_citation&hl=de&user=39qZ_EsAAAAJ&citation_for_view=39qZ_EsAAAAJ:l7tZn2s7bgC

➢ https://images.squarespace-cdn.com/content/v1/596f25c2725e25fb89b3a6f4/1544176431098-WT0ELRS2Q0X1lQClC89YRH/discussion+cc3.0.png?format=1500w