BOAT: Building Auto-Tuners with Structured Bayesian Optimization

Valentin Dalibard  Michael Schaarschmidt  Eiko Yoneki

Presented by Jesse Mu
Parameters in large-scale systems

- Number of cluster nodes
- ML Hyperparams
- Compiler Flags
Parameters in large-scale systems

- Number of cluster nodes
- ML Hyperparams
- Compiler Flags

How to optimize parameters $\theta$?
Parameters in large-scale systems

How to optimize parameters $\theta$?

Minimize some cost function $f(\theta)$.
Parameters in large-scale systems

Coarse

How to optimize parameters \( \theta \)?

Minimize some cost function \( f(\theta) \)

...where cost is runtime, memory, I/O, etc

Fine

Number of cluster nodes

ML Hyperparams

Compiler Flags
Auto-tuning (optimization)
Auto-tuning (optimization)

- Grid search $\theta \in [1, 2, 3, ...]$
Auto-tuning (optimization)

- Grid search $\theta \in [1, 2, 3, ...]$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
Auto-tuning (optimization)

- Grid search $\theta \in [1, 2, 3, \ldots]$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
- Bayesian optimization (e.g. SPEARMINT)
Auto-tuning (optimization) in distributed systems

- Grid search $\theta \in [1, 2, 3, \ldots]$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
- Bayesian optimization (e.g. SPEARMINT)
Auto-tuning (optimization) in distributed systems

- Grid search $\theta \in [1, 2, 3, \ldots]$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
- Bayesian optimization (e.g. SPEARMINT)

Require 1000s of evaluations of cost function!
Auto-tuning (optimization) in distributed systems

- Grid search $\theta \in \{1, 2, 3, \ldots\}$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
- Bayesian optimization (e.g. SPEARMIINT)

Require 1000s of evaluations of cost function!

Fails in high dimensions!
Auto-tuning (optimization) in distributed systems

- Grid search $\theta \in [1, 2, 3, \ldots]$

- Evolutionary approaches (e.g. PetaBricks)

- Hill-climbing (e.g. OpenTuner)

- Bayesian optimization (e.g. SPEARMINT)

- Structured Bayesian optimization (this work: BespOke Auto-Tuners)

Require 1000s of evaluations of cost function!

Fails in high dimensions!
Gaussian Processes

Prior

Data

Posterior

From Carl Rasmussen’s 4F13 lectures
Algorithm 1 The Bayesian optimization methodology

**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: $G \leftarrow G | (x_t, y_t)$

6: end for
Algorithm 1 The Bayesian optimization methodology

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

- e.g. expected increase over max perf.
- (balance exploration vs exploitation)
Bayesian Optimization

1. Configuration Space
2. Objective Function
3. Predicted Performance
Structured Bayesian Optimization (SBO)

1. Configuration Space
2. Objective Function
3. Gaussian Process

Predicted Performance
Performance
Structured Bayesian Optimization (SBO)

1. Configuration Space
2. Objective Function
3. Predicted Performance
Structured Bayesian Optimization (SBO)

*Developer-specified, semi-parametric model of performance from observed performance + arbitrary runtime characteristics*
Structured Bayesian Optimization (SBO)

*Developer-specified, semi-parametric model of performance from observed performance + arbitrary runtime characteristics*
Probabilistic Models for SBO
Probabilistic Models for SBO

(a) Parametric (Linear regression)

(b) Non-parametric (Gaussian process)

(c) Semi-parametric (Combination)
Probabilistic Models for SBO

(a) Parametric (Linear regression)
(b) Non-parametric (Gaussian process)
(c) Semi-parametric (Combination)

Too restrictive  Too generic  Just right
Semi-parametric models in SBO

- Specify the parametric component *only* (GP for free)
Semi-parametric models in SBO

- Specify the parametric component only (GP for free)
- e.g. predict GC rate from JVM *eden* size
Semi-parametric models in SBO

- Specify the parametric component only (GP for free)
- e.g. predict GC rate from JVM eden size

```
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;
    }
};
```
Semi-parametric models in SBO

- Specify the parametric component only (GP for free)
- e.g. predict GC rate from JVM eden size

```cpp
class 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;
    }
};
```

Prior: malloc rate ~ Uniform(0, 5000)
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;
}
Composing semi-parametric models
Composing semi-parametric models
Composing semi-parametric models

Dataflow DAG

Inference exploits conditional independence between models
Composing semi-parametric models

Dataflow DAG

Inference exploits conditional independence between models

```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;
};
```
SBO: Summary

1. Configuration space (i.e. possible params)
2. Objective function + runtime measurements
3. Semi-parametric model of system
SBO: Summary

1. Configuration space (i.e. possible params)
2. Objective function + runtime measurements
3. Semi-parametric model of system
SBO: Summary

1. Configuration space (i.e. possible params)
2. Objective function + runtime measurements
3. Semi-parametric model of system

standard

new
SBO: Summary

1. Configuration space (i.e. possible params)
2. Objective function + runtime measurements
3. *Semi-parametric* model of system

**Key:** try generic system, before optimizing with structure
Evaluation: Cassandra GC
Evaluation: Cassandra GC
Evaluation: Cassandra GC

Best params outperform Cassandra defaults by 63%

Existing systems converge but take 6x longer
Evaluation: Neural Net SGD

Load balancing, worker allocation over 10 machines = 30 params
Evaluation: Neural Net SGD

Load balancing, worker allocation over 10 machines = 30 params
Evaluation: Neural Net SGD

Load balancing, worker allocation over 10 machines = 30 params

Default configuration: 9.82s
OpenTuner: 8.71s
BOAT: 4.31s
Existing systems don’t converge!
Review:
Review: overall, a good, unsurprising contribution
Review: overall, a good, unsurprising contribution

- **Theory**
  - Unsurprising that expert-developed models optimize better!
    - Tradeoff: developer hours vs machine hours
  - Cassandra GC system converges in 2 iterations - model is near-perfect!
    - What happens when parametric model is wrong?
      - More details about tradeoff between parametric model and generic GP
      - OpenTuner: build an ensemble of *multiple* search techniques
Review: overall, a good, unsurprising contribution

● **Theory**
  ○ Unsurprising that expert-developed models optimize better!
    ■ Tradeoff: developer hours vs machine hours
  ○ Cassandra GC system converges in 2 iterations - model is near-perfect!
    What happens when parametric model is wrong?
    ■ More details about tradeoff between parametric model and generic GP
    ■ OpenTuner: build an ensemble of *multiple* search techniques

● **Implementation**
  ○ Cross-validation?
  ○ Key for system adoption: make interface as high-level as possible
Review: overall, a good, unsurprising contribution

● **Theory**
  ○ Unsurprising that expert-developed models optimize better!
    ■ Tradeoff: developer hours vs machine hours
  ○ Cassandra GC system converges in 2 iterations - model is near-perfect!
    What happens when parametric model is wrong?
    ■ More details about tradeoff between parametric model and generic GP
    ■ OpenTuner: build an ensemble of *multiple* search techniques

● **Implementation**
  ○ Cross-validation?
  ○ Key for system adoption: make interface as high-level as possible

● **Evaluation**
  ○ What happens when # params >> 30?
  ○ “DAGModels help debugging”...how?