TrimTuner: Efficient Optimization of Machine Learning Jobs in the Cloud via Sub-Sampling

Pedro Mendes\(^1\), Maria Casimiro\(^{1,2}\), Paolo Romano\(^1\), David Garlan\(^2\)

\(^1\) INESC-ID and Instituto Superior Técnico
\(^2\) Institute of Software Research, Carnegie Mellon University
Contents

1. Motivation
2. TrimTuner
3. Evaluation
4. Final Remarks
Motivation: Training large models on the cloud

- In recent years:
  - Machine Learning (ML) models are increasing in complexity
  - Data-set grow larger and larger

- Training process involves enormous amount of computational resources

- Cloud provides access to “unlimited” on-demand resources to train ML models (and it represents the standard approach nowadays)
Motivation: which cloud and model config. to use?

- Cloud providers offer a large spectrum of Virtual Machines (VMs) with different types, sizes, and prices

- Cloud resources and models hyper-parameters need to be tuned jointly, yielding a vast search space

- Furthermore, testing even a single configuration can be very expensive with current large data-sets
Motivation: joint optimization of cloud & model parameters

- The wrong configuration selection can significantly reduce model’s accuracy and amplify operational costs

- **Disjoint optimization** is unable to identify optimal configurations:
  - up to 3.7x more expensive configurations with Neural Network models [ICDCS20]

Fundamental to **joint optimize** cloud and application parameter
TrimTuner

- System to optimize the training of ML jobs in the cloud

- Exploits data sub-sampling techniques in order to enhance the efficiency of the optimization process

- Jointly optimizes the cloud configuration and hyper-parameters of ML models
TrimTuner - Optimization Problem

\[
\begin{align*}
\text{maximize} \quad & A(x, s = 1) \\
\text{subject to} \quad & q_1(x, s = 1) \geq 0, \ldots, q_m(x, s = 1) \geq 0
\end{align*}
\]

\(x\): configuration \hspace{2cm} \(s\): data-set size [%] \hspace{2cm} \(X\): set of configurations to test

\(A(x,s)\): accuracy of the ML model

\(q_i(x,s)\): Quality of Service (QoS) constraints (e.g., cost or execution time)
TrimTuner

- Deploy the job using **sub-sampled data-sets to reduce the cost** of testing individual configurations

- **Build predictive models** that keep into account how shifts of the data-set size affect the accuracy and cost of the target job

- Use **Gaussian Processes** (GPs) and Ensemble of **Decision Trees** (DTs) as modelling techniques
TrimTuner

- Leverages **Bayesian Optimization** (BO) techniques to update the models and select the next configuration to evaluate

- Does not assume **any a priori knowledge** about the target job and the training platform
Bayesian Optimization: base idea

- Aims at identifying the optimum of an unknown function $f$
- Does not assume any $a\ priori$ knowledge about the target job
- Builds black-box model of $f$:
  
  1. Test a set of initial configurations
  2. Build a model of $f$
  3. $x_i = \arg\max_{x \in \mathcal{X}} \{a(x)\}$; $\mathcal{X}$: untested set; $a$: acquisition function;
  4. Test the configuration $x_i$
  5. Update the model of $f$

**Problem**: testing even a single configuration can be very costly with large data sets
BO using sub-sampling - FABOLAS [AISTATS17]

- Aims to maximize ML model accuracy by optimizing its hyper-parameters

- Leverages BO to solve the optimization problem

- Trains the model with sub-sampled data-sets to reduce the cost of testing configurations
BO + sub-sampling: base idea

- Additional dimension in the search space that corresponds to the data-set size

1. Test a set of initial configurations $x$ in all the sub-sampled data-set sizes
2. Build a model of $f$
3. $(x_i, s_i) = \arg \max_{x \in X, s \in S} \{\alpha(x, s)\}$
   
   $X$: untested set; $S$: sub-sampled data-set sizes; $\alpha$: acquisition function;
4. Test the configuration $(x_i, s_i)$
5. Update the model of $f$
BO + sub-sampling: FABOLAS acquisition function

- **Information gain per unit cost**
  - **Entropy Search** [JMLR12]: estimates the information on the optimum using the full data-set (s=1) that can be gained by testing a sub-sampled configuration \((x,s)\), where \(s<1\).
  - ...normalized by the cost of testing \((x,s)\)

\[
\alpha_F(x, s) = \frac{1}{C(x,s)} \mathbb{E}_{p(y|x,s,S)} \left[ \int p_{opt}^{s=1}(x' | S \cup \{x, s, y\}) \log \frac{p_{opt}^{s=1}(x' | S \cup \{x, s, y\})}{u(x')} \, dx' \right]
\]

Normalize by the cost of testing \((x,s)\)

Entropy Search
FABOLAS: Limitations

1. Does not keep into account any **QoS related constraints**
   - e.g. on model’s training cost or latency.

2. Only optimizes the hyper-parameters of the ML model:
   - Underlooks the problem of **jointly optimizing** cloud parameters
     ⇒ Much smaller search spaces & globally sub-optimal recommendations

3. The acquisition function is **very expensive** to compute:
   - Problem exacerbated with **large search** spaces:
     - i. up to 13 minutes to recommend a configuration (Search space: 1440)
TrimTuner

1. Novel acquisition function to keep into account QoS constraints

2. Optimizes both the hyper-parameters and cloud resources

3. Lightweight models based on Decision Trees (vs Gaussian Processes) Novel filtering heuristic to reduce the number of configurations to compute the acquisition function for.
TrimTuner: acquisition function

- Determines the probability that the new optimum is feasible.
- Estimates the information gain of evaluating \((x,s)\) about the optimum.
- Normalizes by the cost of evaluating \((x,s)\).

\[
\alpha_T(x, s) = \mathbb{E}_{p(q,a|x,s,S)} \left[ \prod_{q_i \in Q} p(q_i(x^*, s = 1) \geq 0 | S \cup \{x, s, q, a\} \right] \alpha_F(x, s)
\]

Need to **simulate the models** using the predicted values.

Probability that the new predicted incumbent meets the constraints

FABOLAS acquisition function
Speeding up the recommendation process

- The computation of FABOLAS acquisition function is very expensive
- TrimTuner tackles this issue by:
  - **Decreasing the number of configurations** for which the acquisition function is evaluated:
    - key idea: filtering non-promising configuration via a fast heuristic
  - Relying on **ensembles of Decision Trees**
    - Unlike FABOLAS, which exploits Gaussian Processes
TrimTuner: filtering out non-promising configurations

- Introduces the Constrained Expected Accuracy (CEA) as the filtering heuristic

\[
CEA(x, s) = A(x, s) \cdot \prod_{q_i \in Q} p(q_i(x, s) \geq 0 | S)
\]

- CEA can be seen as a rough, but cheap, approximation of the acquisition function

- It is computed for all the untested configurations and selects configuration with largest CEA
Evaluation

- Considered baselines:
  - FABOLAS
  - BO using Constrained Expected Improvement (Elc)
  - BO using Elc/USD
  - Random search

- Search space composed of 1440 configurations;

- Target jobs: Distributed training of 3 Neural Networks (NNs) deployed in AWS EC2
Configurations

Cloud parameters

<table>
<thead>
<tr>
<th>VM Type</th>
<th>VM Characteristics</th>
<th>#VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2.small</td>
<td>1 vCPU, 2GB RAM</td>
<td>8, 16, 32, 48, 64, 80</td>
</tr>
<tr>
<td>t2.medium</td>
<td>2 vCPU, 4GB RAM</td>
<td>4, 8, 16, 24, 32, 40</td>
</tr>
<tr>
<td>t2.xlarge</td>
<td>4 vCPU, 16GB RAM</td>
<td>2, 4, 8, 12, 16, 20</td>
</tr>
<tr>
<td>t2.2xlarge</td>
<td>8 vCPU, 32GB RAM</td>
<td>1, 2, 4, 6, 8, 10</td>
</tr>
</tbody>
</table>

Hyper-parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>$10^{-3}, 10^{-4}, 10^{-5}$</td>
</tr>
<tr>
<td>Batch size</td>
<td>16, 256</td>
</tr>
<tr>
<td>Training mode</td>
<td>synchronous, asynchronous</td>
</tr>
<tr>
<td>Data-set size [%]</td>
<td>1/60, 1/10, 1/4, 1/2, 1</td>
</tr>
</tbody>
</table>

Our data-set is publicly available: https://github.com/pedrogbmendes/TrimTuner
## Constraints

Only $\approx 10\%$ of configurations are feasible and have a high accuracy.

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Feasible Configurations</th>
<th>Feasible configurations with high accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>178 (61.8%)</td>
<td>28 (9.72%)</td>
</tr>
<tr>
<td>MLP</td>
<td>161 (55.8%)</td>
<td>29 (10.07%)</td>
</tr>
<tr>
<td>CNN</td>
<td>111 (38.5%)</td>
<td>39 (13.54%)</td>
</tr>
</tbody>
</table>
Evaluation Metrics

- Constrained Accuracy:

\[
Accuracy_C = \begin{cases} 
A(x, s), & \text{if } (x, s) \text{ is feasible} \\
A(x, s) \cdot \frac{C_{\text{max}}}{C(x,s)}, & \text{otherwise}
\end{cases}
\]

- We report the average value of 10 runs.
TrimTuner recommends configurations close to optimum (within 5%) reducing the optimization cost by a factor of 50x w.r.t. Elc.

Initial sampling: dashed lines

Better
TrimTuner Gains

Cost savings

Time speed ups

Cost reduced by up to:
- \(50x\) w.r.t. Elc
- \(\approx 10x\) w.r.t Elc/USD

Time accelerated by up to:
- \(65x\) w.r.t. Elc
- \(15x\) w.r.t Elc/USD
Recommendation Times

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Avg. time to recommend a configuration [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrimTuner (GPs)</td>
<td>18.65</td>
</tr>
<tr>
<td>TrimTuner (DTs)</td>
<td>1.36</td>
</tr>
<tr>
<td>Fabolas</td>
<td>13.96</td>
</tr>
<tr>
<td>Elc (Elc/USD)</td>
<td>1.17</td>
</tr>
</tbody>
</table>

Recommendation time reduced by **13.7x**
Efficiency of the CEA

1. CEA compared with other filtering heuristics:
   - Direct [JGO01]
   - CMAES [EC03]
   - Random approach

   Two state-of-the-art black-box optimizers
Efficiency of the CEA

CEA reduces the recommendation cost by up to \(7x/3.6x\) w.r.t. Direct/CMAES

<table>
<thead>
<tr>
<th>Filtering Heuristic</th>
<th>Rec. Time TT DTs [min]</th>
<th>Rec. Time TT GPs [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEA</td>
<td>1.72</td>
<td>16.85</td>
</tr>
<tr>
<td>Direct</td>
<td>2.63</td>
<td>36.18</td>
</tr>
<tr>
<td>CMAES</td>
<td>2.26</td>
<td>30.87</td>
</tr>
<tr>
<td>Random</td>
<td>1.62</td>
<td>16.53</td>
</tr>
</tbody>
</table>
Final Remarks

- TrimTuner: novel system to **optimize ML model training in the cloud**
  - Maximizes model accuracy subject to QoS constraint
  - Exploits sub-sampling data to reduce the costs

- TrimTuner relies on subsampling techniques to reduce the:
  - Training cost by up to **50x**,  
  - Latency of the exploration process by up to **65x** (compared with BO using EIc);

- It accelerates the recommendation process by up to **117x** via
  - Novel filtering heuristic: CEA;
  - Lightweight predictive models based on DTs.
Code and data-sets are available online

https://github.com/pedrogbmendes/TrimTuner

Check our paper for more details