SLAQ: Quality-Driven Scheduling for Distributed Machine Learning

Haoyu Zhang*, Logan Stafman*, Andrew Or, Michael J. Freedman
“AI is the new electricity.”

- Machine translation
- Recommendation system
- Autonomous driving
- Object detection and recognition

Supervised Learning
Unsupervised Learning
Transfer Learning
Reinforcement Learning
ML algorithms are *approximate*

- ML model: a parametric transformation

$$f_\theta$$
ML algorithms are *approximate*

- ML model: a parametric transformation

  \[ X \xrightarrow{f_\theta} Y \]

- maps input variables $X$ to output variables $Y$
- typically contains a set of parameters $\theta$

- **Quality**: how well model maps input to the correct output
- **Loss function**: discrepancy of model output and ground truth
Training ML models: an iterative process

- Training algorithms iteratively minimize a loss function
  - E.g., stochastic gradient descent (SGD), L-BFGS
Training ML models: an *iterative* process

- Quality improvement is subject to **diminishing returns**
  - More than **80% of work done in 20% of time**
Exploratory ML training: not a one-time effort

- Train model multiple times for exploratory purposes
- Provide early feedback, direct model search for high quality models
How to schedule multiple training jobs on shared cluster?

- Key features of ML jobs
  - Approximate
  - Diminishing returns
  - Exploratory process

- Problem with resource fairness scheduling
  - Jobs in early stage: could benefit a lot from additional resources
  - Jobs almost converged: make only marginal improvement
SLAQ: quality-aware scheduling

• Intuition: in the context of approximate ML training, more resources should be allocated to jobs that have the most potential for quality improvement
Solution Overview

- Normalize quality metrics
- Predict quality improvement
- Quality-driven scheduling
## Normalizing quality metrics

<table>
<thead>
<tr>
<th></th>
<th>Applicable to All Algorithms?</th>
<th>Comparable Magnitudes?</th>
<th>Known Range?</th>
<th>Predictable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy / F1 Score / Area Under Curve / Confusion Matrix / etc.</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Loss</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Normalized Loss</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>$\Delta$Loss</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Normalized $\Delta$Loss</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Normalizing quality metrics

• Normalize change of loss values w.r.t. largest change so far
  • Currently does not support some non-convex optimization algorithms
Training iterations: loss prediction

- Previous work: offline profiling / analysis [Ernest NSDI 16] [CherryPick NSDI 17]
  - Overhead for frequent offline analysis is huge
- Strawman: use last $\Delta$Loss as prediction for future $\Delta$Loss
- SLAQ: online prediction using weighted curve fitting
Scheduling approximate ML training jobs

- Predict how much quality can be improved when assign X workers to jobs
- Reassign workers to maximize quality improvement
Experiment setup

- Representative mix of training jobs with
- Compare against a work-conserving fair scheduler

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Acronym</th>
<th>Type</th>
<th>Optimization Algorithm</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means</td>
<td>K-Means</td>
<td>Clustering</td>
<td>Lloyd Algorithm</td>
<td>Synthetic</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>LogReg</td>
<td>Classification</td>
<td>Gradient Descent</td>
<td>Epsilon [33]</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>SVM</td>
<td>Classification</td>
<td>Gradient Descent</td>
<td>Epsilon</td>
</tr>
<tr>
<td>SVM (polynomial kernel)</td>
<td>SVMPoly</td>
<td>Classification</td>
<td>Gradient Descent</td>
<td>MNIST [34]</td>
</tr>
<tr>
<td>Gradient Boosted Tree</td>
<td>GBT</td>
<td>Classification</td>
<td>Gradient Boosting</td>
<td>Epsilon</td>
</tr>
<tr>
<td>GBT Regression</td>
<td>GBTReg</td>
<td>Regression</td>
<td>Gradient Boosting</td>
<td>YearPredictionMSD [35]</td>
</tr>
<tr>
<td>Multi-Layer Perceptron Classifier</td>
<td>MLPC</td>
<td>Classification</td>
<td>L-BFGS</td>
<td>Epsilon</td>
</tr>
<tr>
<td>Latent Dirichlet Allocation</td>
<td>LDA</td>
<td>Clustering</td>
<td>EM / Online Algorithm</td>
<td>Associated Press Corpus [36]</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>LinReg</td>
<td>Regression</td>
<td>L-BFGS</td>
<td>YearPredictionMSD</td>
</tr>
</tbody>
</table>
Evaluation: resource allocation across jobs

- 160 training jobs submitted to cluster following Poisson distribution
  - 25% jobs with high loss values
  - 25% jobs with medium loss values
  - 50% jobs with low loss values (almost converged)
Evaluation: cluster-wide quality and time

Quality

• SLAQ’s average loss is 73% lower than that of the fair scheduler

Time

• SLAQ reduces time to reach 90% (95%) loss reduction by 45% (30%)
SLAQ Evaluation: Scalability

- Frequently reschedule and reconfigure in reaction to changes of progress
- Even with thousands of concurrent jobs, SLAQ makes rescheduling decisions in just a few seconds
Conclusion

• SLAQ leverages the approximate and iterative ML training process

• Highly tailored prediction for iterative job quality

• Allocate resources to maximize quality improvement

• SLAQ achieves better overall quality and end-to-end training time
Training iterations: runtime prediction

- Iteration runtime: $c \cdot S/N$
  - Model complexity $c$, data size $S$, number of workers $N$
  - Model update (i.e., size of $\Delta \theta$) is comparably much smaller