Neural Network
Hyperparameter Optimization

Chris Ouyang (CUHK Mathematics)  Daniel McBride (UTK Mathematics)
Presentation Outline

- **Introduction**

- **Part I:** An Early Stopping Algorithm Based on Learning Curve Matching
  Chris Ouyang

- **Part II:** Population Based Training with MagmaDNN and OpenDIEL
  Daniel McBride
Introduction

● What is a hyperparameter?

They are neural network “presets” like network architecture, learning rate, batch size, and more.

● Why do we need to optimize the hyperparameters?

A poor choice of hyperparameters can cause a network’s accuracy to converge slowly or not at all.
Introduction

- What are some obstacles to optimizing hyperparameters?
  - The Curse of Dimensionality
  - Highly irregular (nonconvex, nondifferentiable) search spaces

- What are some standard hyperparameter optimization techniques?
  - Classic Approaches: Grid Search, Random Search
  - Modern Approaches: Early Stopping, Evolutionary Algorithms
Part I
An Early Stopping Algorithm
Based on Learning Curve Matching

Chris Ouyang
Hyperparameter Algorithms

- **Hyperparameter Selection:** Random search, grid search and Bayesian optimization
- **Early stopping:** Successive Halving Algorithm (SHA) and Hyperband
- **Advanced Algorithm:** Evolutionary Algorithm, such as population based training (PBT) and swarm optimization.
**LCM Algorithm: Flow Chart and Terms**

- **Trials**: Sets contain a single sample for every hyperparameter.
- **Learning Curves**: Arrays of the numerical values of loss function in some certain stages during a single training.
- **Check Points**: Points where apply LCM to decide whether to abort the training.
LCM Algorithm: Cumulation Stage

Data Set:
[LC_1, Performance_1],
[LC_2, Performance_2],
[LC_3, Performance_3],
[LC_4, Performance_4],
……
[LC_m, Performance_m]

Learning Curve with performance
[Loss_1, Loss_2, Loss_3, Loss_4, Loss_5, …… , Loss_n, Performance]
LCM Algorithm: Checking Stage

Data Set:
[L11, L12, L13, ..., L1k],
[L21, L22, L23, ..., L2k],
[L31, L32, L33, ..., L3k],
[L41, L42, L43, ..., L4k],
........
[Lm1, Lm2, Lm3, ..., Lmk]

Learning Curve
[Loss_1, Loss_2, Loss_3, ...., Loss_k]

Data Set:
[L11, L12, L13, ..., L1n Performance_1],
[L21, L22, L23, ..., L2n Performance_2],
[L31, L32, L33, ..., L3n Performance_3],
[L41, L42, L43, ..., L4n Performance_4],
........
[Lm1, Lm2, Lm3, ..., Lmn, Performance_m]

An Early Stopping Algorithm Based on Learning Curve Matching - Chris Ouyang
**LCM Algorithm: Checking Stage**

---

**Distance Function F**
(such as L1, L2, L\(\infty\))

---

**Learning Curve**
[L\(_1\), L\(_2\), L\(_3\), \ldots, L\(_k\)]

---

**Data Set:**
[L\(_11\), L\(_12\), L\(_13\), \ldots, L\(_1k\)],
[L\(_21\), L\(_22\), L\(_23\), \ldots, L\(_2k\)],
[L\(_31\), L\(_32\), L\(_33\), \ldots, L\(_3k\)],
[L\(_41\), L\(_42\), L\(_43\), \ldots, L\(_4k\)],
\ldots
[L\(_m1\), L\(_m2\), L\(_m3\), \ldots, L\(_mk\)]

---

**Distance Function F**
(such as L1, L2, L\(\infty\))

---

**Distance List:**
[Distance\(_1\),
Distance\(_2\),
Distance\(_3\),
Distance\(_4\)
\ldots
Distance\(_m\)]

---

**Data Set:**
[Performance\(_1\),
Performance\(_2\),
Performance\(_3\),
Performance\(_4\),
\ldots,
Performance\(_m\)]

---

**Predicted Performance:**
Performance\(_j\)

---

The rank of it: rank\(_j\)
The rank percentage: Percentage\(_j\)

---

If Per\(_j\) < keep_rate:
Stop training;
Otherwise, continue training.
LCM Algorithm: Comparisons

- **Network**: Only one dense layer
- **Dataset**: MNIST
- **Optimizer**: Stochastic gradient descent
- **Hyperparameter**: Epochs, batch sizes, learning rate, momentum and decay
- **Benchmark**: Random search
- **Times**: 9

<table>
<thead>
<tr>
<th></th>
<th>Trials</th>
<th>Computer Time (S)</th>
<th>Best Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCM</td>
<td>100</td>
<td>778.50</td>
<td>97.10</td>
</tr>
<tr>
<td>Random</td>
<td>100</td>
<td>3657.75</td>
<td>97.41</td>
</tr>
</tbody>
</table>

**Remark**: In 5 of 9 experiments, two algorithms got the same optimal hyperparameters.
### LCM Algorithm: Comparisons

- **Network**: Only one dense layer
- **Dataset**: MNIST
- **Optimizer**: Stochastic gradient descent
- **Hyperparameter**: Epochs, batch sizes, learning rate, momentum and decay
- **Benchmark**: Random search
- **Times**: 6

<table>
<thead>
<tr>
<th></th>
<th>Trials</th>
<th>Computer Time (S)</th>
<th>Best Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCM</td>
<td>37.67</td>
<td>4800</td>
<td>97.82</td>
</tr>
<tr>
<td>Random</td>
<td>67.33</td>
<td>4800</td>
<td>97.69</td>
</tr>
</tbody>
</table>

**Remark**: In 4 of 6 experiments, two algorithms got the same optimal hyperparameters.
LCM Algorithm: Comparisons

- **Network**: Four CNN layers and several dense layers
- **Dataset**: CIFAR10
- **Optimizer**: Adam
- **Hyperparameter**: More than 10 hyperparameters
- **Benchmark**: Random search
- **Times**: 12

<table>
<thead>
<tr>
<th></th>
<th>Trials</th>
<th>Computer Time (S)</th>
<th>Best Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCM</td>
<td>100</td>
<td>8069.08</td>
<td>67.05</td>
</tr>
<tr>
<td>Random</td>
<td>100</td>
<td>26498.00</td>
<td>67.26</td>
</tr>
</tbody>
</table>

**Remark**: in 7 of 12 experiments, two algorithms got the same optimal hyperparameters.
Part II
Population Based Training with MagmaDNN and OpenDIEL

Daniel McBride
PBT: Background

- **What is Population Based Training (PBT)?**

  PBT is an evolutionary hyperparameter optimization algorithm.

- **Evolutionary optimization algorithms** use natural models to inspire a particular approach to traversing a search space. One classic case is the Particle Swarm Optimization algorithm, inspired by the swarming behavior of bees.
PBT: Background

- **What are the benefits of PBT?**

  PBT outperforms the standard hyperparameter tuning benchmarks. These benchmark algorithms, **Grid Search and Random Search**, each have their own limitations, which PBT overcomes.

- **Why should we implement it on MagmaDNN and OpenDIEL?**
  - MagmaDNN and OpenDIEL are engineered for supercomputers.
  - The current standard implementation (Ray-Tune: shared memory model) has a scalability bottleneck.
PBT: Algorithm

How does the PBT Algorithm work?

- Population Model
- Stochasticity
- Explore / Exploit
- Early Stopping
- Evolution
- Adaptive Hyperparameter Scheduling
**PBT:** Algorithm

How does the PBT Algorithm work?
# PBT: Algorithm

## Does PBT’s functionality improve on the benchmark algorithms?

<table>
<thead>
<tr>
<th></th>
<th>Grid Search</th>
<th>Random Search</th>
<th>PBT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelizability</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Stochasticity</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Early Stopping</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Adaptive Hyperparameters</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
PBT: Analysis - Dynamic Learning Rate

- **Data: MNIST**
  - 60k images of handwritten digits 0-9
  - 256 greyscale pixels per image
  - 10 categories (0-9)

- **Network: MagmaDNN**
  - Network Structure: In -> FCB -> Sig -> FCB -> Sig -> FCB -> Out
  - Weight Optimizer: Stochastic Gradient Descent
  - Number of Epochs = 5
  - Batch Size = 32

- **Benchmark**: constant learning rate = .0016

- **Experiments**: dynamic learning rate schedules with variable initial values

*FCB := Fully Connected Layer with Bias
*Sig := Sigmoid Activation
PBT: Analysis - Dynamic Learning Rate

MNIST 60k MagmaDNN: Dynamic vs Static Learning Rate - LEARNING RATE SCHEDULES

MNIST 60k MagmaDNN: Dynamic vs Static Learning Rate - FINAL ACCURACY

Population Based Training with MagmaDNN and OpenDIEL - Daniel McBride
PBT: Goals

- Extend the OpenDIEL Grid Search Application to have PBT functionality, i.e. stochasticity and evolution.
- Program more custom MagmaDNN classes to explore the effect of tuning Convolutional Neural Network hyperparameters.
- Implement PBT on MagmaDNN and OpenDIEL with a distributed Worker, and overcome the Ray-Tune bottleneck.
Thanks for listening!
- The Hyperparameter Team