BAYESIAN GLOBAL OPTIMIZATION
Using Optimal Learning to Tune Deep Learning Pipelines

Scott Clark
scott@sigopt.com
OUTLINE

1. Why is Tuning AI Models Hard?
2. Comparison of Tuning Methods
3. Bayesian Global Optimization
4. Deep Learning Examples
5. Evaluating Optimization Strategies
Deep Learning / AI is extremely powerful

Tuning these systems is extremely non-intuitive
What is the most important unresolved problem in machine learning?

“...we still don't really know why some configurations of deep neural networks work in some case and not others, let alone having a more or less automatic approach to determining the architectures and the hyperparameters.”

Xavier Amatriain, VP Engineering at Quora
(former Director of Research at Netflix)
TUNABLE PARAMETERS IN DEEP LEARNING

**Data**
- Which dataset do you want to use?
  - [Image of dataset options]
- Ratio of training to test data: 60%
- Noise: 0
- Batch size: 10

**Input**
- Which properties do you want to feed in?
  - $X_1$, $X_2$, $X_1^2$, $X_2^2$, $X_1X_2$, $\sin(X_1)$, $\sin(X_2)$

**Hidden Layers**
- 4 hidden layers
- 4 neurons
- 2 neurons
- 7 neurons
- 2 neurons

**Output**
- Test loss: 0.391
- Training loss: 0.218
- The outputs are mixed with varying weights, shown by the thickness of the lines.

Colors show data, neuron, and weight values.
class mxnet.optimizer. RMSProp (learning_rate=0.001, gamma1=0.9, gamma2=0.9, epsilon=1e-08, centered=False, clip_weights=None, **kwargs)

The RMSProp optimizer.

Two versions of RMSProp are implemented:


If centered=True, we follow http://arxiv.org/pdf/1308.0850v5.pdf (38)-(45) by Alex Graves, 2013.

This optimizer accepts the following parameters in addition to those accepted by Optimizer:

Parameters:

- **gamma1** (float, optional) – Decay factor of moving average for gradient^2.
- **gamma2** (float, optional) – A “momentum” factor. Only used if centered=True.
- **epsilon** (float, optional) – Small value to avoid division by 0.
- **centered** (bool, optional) – Use Graves’ or Tieleman & Hinton’s version of RMSProp.
- **clip_weights** (float, optional) – clip weights into range [-clip_weights, clip_weights]
STANDARD METHODS
FOR HYPERPARAMETER SEARCH
STANDARD TUNING METHODS

Trainig Data → ML / AI Model → Cross Validation → Parameter Configuration
- Weights
- Thresholds
- Window sizes
- Transformations

Manual Search
Grid Search
Random Search
OPTIMIZATION FEEDBACK LOOP

- Training Data
- Testing Data
- ML / AI Model
- Cross Validation
- New configurations
- Better Results
- Objective Metric
- REST API
BAYESIAN GLOBAL OPTIMIZATION
… the challenge of how to collect information as efficiently as possible, primarily for settings where collecting information is time consuming and expensive.

Prof. Warren Powell - Princeton

What is the most efficient way to collect information?

Prof. Peter Frazier - Cornell

How do we make the most money, as fast as possible?

Scott Clark - CEO, SigOpt
BAYESIAN GLOBAL OPTIMIZATION

- Optimize objective function
  - Loss, Accuracy, Likelihood
- Given parameters
  - Hyperparameters, feature/architecture params
- Find the best hyperparameters
  - Sample function as few times as possible
  - Training on big data is expensive
HOW DOES IT WORK?

SMBO

Sequential Model-Based Optimization
1. Build Gaussian Process (GP) with points sampled so far
2. Optimize the fit of the GP (covariance hyperparameters)
3. Find the point(s) of highest Expected Improvement within parameter domain
4. Return optimal next best point(s) to sample
GAUSSIAN PROCESSES
GAUSSIAN PROCESSES
GAUSSIAN PROCESSES
GAUSSIAN PROCESSES
GAUSSIAN PROCESSES

(a), prior

(b), posterior
EXPECTED IMPROVEMENT
EXPECTED IMPROVEMENT
EXPECTED IMPROVEMENT
EXPECTED IMPROVEMENT
DEEP LEARNING EXAMPLES
Classify movie reviews using a CNN in MXNet
TEXT CLASSIFICATION PIPELINE

Training Text

ML / AI Model (MXNet)

Validation

Better Results

Accuracy

Hyperparameter Configurations and Feature Transformations

SIGOPT

SIGOPT REST API

Testing Text
class mxnet.optimizer. RMSProp(learning_rate=0.001, gamma1=0.9, gamma2=0.9, epsilon=1e-08, centered=False, clip_weights=None, **kwargs)

The RMSProp optimizer.

Two versions of RMSProp are implemented:


If centered=True, we follow http://arxiv.org/pdf/1308.0850v5.pdf (38)-(45) by Alex Graves, 2013.

This optimizer accepts the following parameters in addition to those accepted by Optimizer:

Parameters:

- **gamma1** (float, optional) – Decay factor of moving average for \( \text{gradient}^2 \).
- **gamma2** (float, optional) – A “momentum” factor. Only used if centered=True.
- **epsilon** (float, optional) – Small value to avoid division by 0.
- **centered** (bool, optional) – Use Graves’ or Tieleman & Hinton’s version of RMSProp.
- **clip_weights** (float, optional) – clip weights into range \([-\text{clip_weights}, \text{clip_weights}]\)
Comparison of several RMSProp SGD parametrizations
ARCHITECTURE PARAMETERS

wait for the video and don't rent it

$n \times k$ representation of sentence with static and non-static channels

Convolutional layer with multiple filter widths and feature maps

Max-over-time pooling

Fully connected layer with dropout and softmax output
TUNING METHODS

Grid Search

Random Search

SIGOPT
MULTIPLICATIVE TUNING SPEED UP

Provides a 400x speed up
**SPEED UP #1: CPU -> GPU**

<table>
<thead>
<tr>
<th>NVIDIA GPU</th>
<th>vCPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 seconds per epoch</td>
<td>146 seconds per epoch</td>
</tr>
</tbody>
</table>

- optimizer rmsprop
- maximum gradient 5.0
- learning rate (step size) 0.0005
- epochs to train for 50

| Iter [0] Train: Time: 3.212s |

| Iter [0] Train: Time: 146.908s |
| Iter [1] Train: Time: 146.520s |
## Speed Up #2: Random/Grid -> SigOpt

<table>
<thead>
<tr>
<th></th>
<th>SigOpt</th>
<th>Random Search</th>
<th>Grid Search</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic</strong></td>
<td>240 trials</td>
<td>2400 trials</td>
<td>729 trials</td>
</tr>
<tr>
<td><strong>Complex</strong></td>
<td>400 trials</td>
<td>4000 trials</td>
<td>59049 trials</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SigOpt</th>
<th>Random Search</th>
<th>Grid Search</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relative Performance</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Basic Scenario</strong></td>
<td>+6.2%</td>
<td>+5.6%</td>
<td>+4.8%</td>
</tr>
<tr>
<td><strong>Complex Scenario</strong></td>
<td>+7.0%</td>
<td>+5.8%</td>
<td>Not feasible</td>
</tr>
</tbody>
</table>
CONSISTENTLY BETTER AND FASTER

Interquartile Progression

Accuracy

Number of observations

- SigOpt
- Random
- Grid
Classify house numbers in an image dataset (SVHN)
COMPUTER VISION PIPELINE

Training Images

ML / AI Model (Tensorflow)

Testing Images

Cross Validation

Hyperparameter Configurations and Feature Transformations

Better Results

Accuracy

Σ SIGOPT

Σ SIGOPT REST API
All convolutional neural network
Multiple convolutional and dropout layers
Hyperparameter optimization mixture of domain expertise and grid search (brute force)

COMPARATIVE PERFORMANCE

- Expert baseline: 0.8995
  - (using neon)
- SigOpt best: 0.9011
  - 1.6% reduction in error rate
  - No expert time wasted in tuning
Explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions

- Variable depth
- Hyperparameter optimization mixture of domain expertise and grid search (brute force)

COMPARATIVE PERFORMANCE

- Expert baseline: 0.9339
  - (from paper)

- SigOpt best: 0.9436
  - 15% relative error rate reduction
  - No expert time wasted in tuning
EVALUATING THE OPTIMIZER
What is the best value found after optimization completes?

<table>
<thead>
<tr>
<th>METRIC: BEST FOUND</th>
<th>BLUE</th>
<th>RED</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEST_FOUND</td>
<td>0.7225</td>
<td>0.8949</td>
</tr>
</tbody>
</table>
How quickly is optimum found? (area under curve)

<table>
<thead>
<tr>
<th></th>
<th>BLUE</th>
<th>RED</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEST_FOUND</td>
<td>0.9439</td>
<td>0.9435</td>
</tr>
<tr>
<td>AUC</td>
<td>0.8299</td>
<td>0.9358</td>
</tr>
</tbody>
</table>
STOCHASTIC OPTIMIZATION

<table>
<thead>
<tr>
<th>Metric</th>
<th>PSO</th>
<th>2015_11_30_22</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM_BEST_FOUND+</td>
<td>0.6284319023543677</td>
<td>0.8006288853893207</td>
<td>0.0004604568971440039</td>
</tr>
<tr>
<td>NORM_AUC+</td>
<td>0.37554939397593576</td>
<td>0.5126848207184765</td>
<td>0.00183119320111959</td>
</tr>
</tbody>
</table>
- Optimization functions from literature
- ML datasets: LIBSVM, Deep Learning, etc

<table>
<thead>
<tr>
<th>TEST FUNCTION TYPE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Params</td>
<td>184</td>
</tr>
<tr>
<td>Noisy Observations</td>
<td>188</td>
</tr>
<tr>
<td>Parallel Observations</td>
<td>45</td>
</tr>
<tr>
<td>Integer Params</td>
<td>34</td>
</tr>
<tr>
<td>Categorical Params / ML</td>
<td>47</td>
</tr>
<tr>
<td>Failure Observations</td>
<td>30</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>489</strong></td>
</tr>
</tbody>
</table>
On-demand cluster in AWS for parallel eval function optimization

Full eval consists of ~20000 optimizations, taking ~30 min
1. Mann-Whitney U tests using BEST_FOUND
2. Tied results then partially ranked using AUC
3. Any remaining ties, stay as ties for final ranking
RANKING AGGREGATION

- Aggregate partial rankings across all eval functions using Borda count (sum of methods ranked lower)

\[
\begin{align*}
  f_1 &: \ A > B > C > D, \\
  f_2 &: \ (A, B) > C, D, \\
  f_3 &: \ C > A > B, D, \\
  f_4 &: \ D > (A, C) > B, \\
  f_5 &: \ (A, B, C, D) > 0, \\
  f_6 &: \ B > (A, C, D),
\end{align*}
\]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Borda</th>
<th>Firsts</th>
<th>Top Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method A</td>
<td>8</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Method B</td>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Method C</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Method D</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
## SHORT RESULTS SUMMARY

### Table 8. Noisy functions (24 total)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Borda</th>
<th>Firsts</th>
<th>Top Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>SigOpt</td>
<td>89</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>Spearmint</td>
<td>60</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>HyperOpt</td>
<td>47</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>SMAC</td>
<td>13</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>PSO</td>
<td>64</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>Grid</td>
<td>22</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Random</td>
<td>16</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

### Table 6. Mostly boring functions (10 total)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Borda</th>
<th>Firsts</th>
<th>Top Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spearmint</td>
<td>46</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>SigOpt</td>
<td>45</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>HyperOpt</td>
<td>33</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>SMAC</td>
<td>11</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>PSO</td>
<td>21</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Grid</td>
<td>8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Random</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 7. Boundary optimum functions (13 total)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Borda</th>
<th>Firsts</th>
<th>Top Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spearmint</td>
<td>74</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>SigOpt</td>
<td>67</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>HyperOpt</td>
<td>29</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>SMAC</td>
<td>7</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>PSO</td>
<td>30</td>
<td>-</td>
<td>11</td>
</tr>
<tr>
<td>Grid</td>
<td>14</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Random</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 5. >10D functions (8 total)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Borda</th>
<th>Firsts</th>
<th>Top Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>SigOpt</td>
<td>40</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>HyperOpt</td>
<td>35</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Spearmint</td>
<td>23</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SMAC</td>
<td>12</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>Random</td>
<td>16</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>PSO</td>
<td>12</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>Grid</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Diagrams

- [Diagram 1](image1.png)
- [Diagram 2](image2.png)
SIGOPT SERVICE
OPTIMIZATION FEEDBACK LOOP

- **Training Data**
- **Testing Data**
- **ML / AI Model**
- **Cross Validation**
- **Better Results**
- **Objective Metric**
- **New configurations**

**SIGOPT REST API**
SIMPLIFIED OPTIMIZATION

Client Libraries
- Python
- Java
- R
- Matlab
- And more...

Framework Integrations
- TensorFlow
- scikit-learn
- xgboost
- Keras
- Neon
- And more...

Live Demo
DISTRIBUTED TRAINING

- SigOpt serves as a distributed scheduler for training models across workers
- Workers access the SigOpt API for the latest parameters to try for each model
- Enables easy distributed training of non-distributed algorithms across any number of models
COMPARATIVE PERFORMANCE

- **Better Results, Faster and Cheaper**
  Quickly get the most out of your models with our proven, peer-reviewed ensemble of Bayesian and Global Optimization Methods
  - A Stratified Analysis of Bayesian Optimization Methods (ICML 2016)
  - Evaluation System for a Bayesian Optimization Service (ICML 2016)
  - Interactive Preference Learning of Utility Functions for Multi-Objective Optimization (NIPS 2016)
  - And more...

- **Fully Featured**
  Tune any model in any pipeline
  - Scales to 100 continuous, integer, and categorical parameters and many thousands of evaluations
  - Parallel tuning support across any number of models
  - Simple integrations with many languages and libraries
  - Powerful dashboards for introspecting your models and optimization
  - Advanced features like multi-objective optimization, failure region support, and more

- **Secure Black Box Optimization**
  Your data and models never leave your system
Try it yourself!

https://sigopt.com/getstarted
Questions?

contact@sigopt.com

SIGOPT
https://sigopt.com
@SigOpt