Hyperparameter Tuning in Kubeflow

Richard Liu (Senior Software Engineer, Google)
ricliu@google.com | @richardsliu

Johnu George (Technical Lead, Cisco)
johnugeo@cisco.com | @johnugeorge

June 25, 2019
Agenda

- Hyperparameter Tuning - What it is and why it is hard
- Kubeflow and Katib
- System Architecture
- Demo
- Neural Architecture Search
- Future Work
An Example: Digits Recognition with MNist

```python
fit.add_fit_args(parser)
parser.set_defaults(
    network = 'mlp',
    # train
    gpus = None,
    batch_size = 64,
    disp_batches = 100,
    num_epochs = 20,
    lr = .05,
    lr_step_epochs = '10'
)
args = parser.parse_args()

# load network
from importlib import import_module
net = import_module('symbols.'+args.network)
sym = net.get_symbol(**vars(args))

# train
fit.fit(args, sym, get_mnist_iter)
```

Source: [https://github.com/apache/incubator-mxnet/blob/master/example/image-classification/train_mnist.py](https://github.com/apache/incubator-mxnet/blob/master/example/image-classification/train_mnist.py)
What is Hyperparameter Tuning?

- **Hyperparameters**: Configuration variables that are external to the model, set before the training process begins
  - Ex: Batch size, learning rate
- Setting the right hyperparameters can significantly improve your model performance
- ... but only if done correctly, which is hard
- **Hyperparameter Tuning**: Finding values for hyperparameters that optimizes an objective function
  - Ex: Finding the optimal batch size and learning rate to maximize prediction accuracy
Why is Hyperparameter Tuning Hard?

- More hyperparameters -> exponential search space growth
- Tuning by hand is inefficient and error-prone
- Need to tracking metrics across multiple jobs
- Managing resources and infrastructure for lots of jobs is hard
- Variety of frameworks and algorithms to support
How does Kubernetes Help?

- Microservice architecture -> simple to build self-contained, lightweight services
- Containerization -> increased resilience and scalability
- Declarative API -> straightforward to describe the desired state, makes managing resources simple
- Flexible API -> custom resource definition allows users to interact with objects using standard REST APIs and kubectl
- Portability -> go from local development to on-prem hosting to cloud
Introducing Kubeflow

- A Kubernetes-native ML platform for developing, orchestrating, deploying, and running scalable end-to-end ML workloads
- Make deployments of ML simple, portable, and scalable

Katib: Hyperparameter Tuning in Kubeflow

- Inspired by Google Vizier(*)
- Fully open-source: [https://github.com/kubeflow/katib](https://github.com/kubeflow/katib)
- Framework agnostic
  - TensorFlow
  - PyTorch
  - MxNet
- Customizable algorithms
  - Random search
  - Grid search
  - Bayesian optimization
  - Hyperband

Concepts: Experiment

- **Experiment**: an end-to-end process for HP optimization. E.g.:
  - Finding hyperparameter values for a digits recognition model

- **An Experiment has**...
  - **Objective**: What we are trying to optimize
  - **Search Space**: Constraints for configurations
  - **Search Algorithm**: How to find the optimal configurations

- **Experiment is a Custom Resource**
  - Allows standard k8s APIs
  - Can use kubectl to interact
  - State is stored in etcd
  - Lifecycle managed by controllers
Concepts: Suggestion

- **Suggestion:** a proposed solution to the optimization problem
  - E.g. one set of hyperparameter values
- Each suggestion algorithm is a standalone microservice
  - Allows users to create customized suggestion algorithms
- *Experiment controller* contacts *Suggestion service* to get new configurations for *Trials*
Concepts: Trial

- Trial: one iteration of the optimization process.
  - E.g. one instance of a training job, using one set of HPs
- A Trial has:
  - A set of specific parameter assignments
  - A “worker” process that runs the trial in a container
  - Observation metrics - how did we do?
- Trial is an internal Custom Resource
  - Experiment controller spawns/manages Trials
  - Each Trial runs in a Docker container
  - Can scale up for distributed training
Workflow for Hyperparameter Tuning

# Initialize search space
# Initialize model

while not objective_reached and not budget_exhausted:
    # Obtain the next set of hyperparameters
    hyperparameters = GetSuggestions()

    # Collect metrics
    metrics = RunTrial(hyperparameters)

    # Report metrics
    ReportMetrics(metrics)
System Architecture
Demo
Classical vs Automated Machine Learning

- Classical machine learning: human experts...
  - Select features
  - Choose algorithm
  - Configure hyperparameters
  - Evaluate performance
  - Tune models
- Automated machine learning:
  - A program generates the model without human intervention
Landscape of Automated Machine Learning

Source: https://github.com/hibayesian/awesome-automl-papers
Neural Architecture Search

- Algorithm may search for an optimal network, or search for optimal cell (subgraph)
- Evolve strategy can be by generation or by modification
# Initialize search space
# Initialize neural network

```python
while not objective_reached and not budget_exhausted:
    # Obtain the next set of operations
    operations = GetSuggestions()

    # Construct model
    model = ConstructModel(operations)

    # Collect metrics
    metrics = RunTrial(model)

    # Report metrics
    ReportMetrics(metrics)
```
What’s Coming?

- Better production support
  - Support for customizable database backend
  - Metadata store integration
  - Support for long-running experiments
- More features for automated machine learning
  - Model compression
  - Automated feature engineering
How to Contribute?

- GitHub: [https://github.com/kubeflow/katib](https://github.com/kubeflow/katib)
  - Feedback and feature requests
  - “Help Wanted” features
  - New algorithms
  - Infrastructure and testing improvements
- Invitation to our Slack channel
- Mailing list: kubeflow-discuss
Thank You

Yuji Oshima, NTT
Hougang Liu, IBM
Jinan Zhou, Cisco
Anubhav Garg, Cisco

Ce Gao, Caicloud
Guangya Liu, IBM
Andrey Velichkevich, Cisco
Kirill Prosvirov, Cisco
## Demo: Setting Up an Experiment

### YAML File

<table>
<thead>
<tr>
<th>Metadata</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>mnist-experiment</td>
</tr>
<tr>
<td>Namespace</td>
<td></td>
<td>kubeflow</td>
</tr>
</tbody>
</table>

### Common Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParallelTrialCount</td>
<td>10</td>
</tr>
<tr>
<td>MaxTrialCount</td>
<td>100</td>
</tr>
<tr>
<td>MaxFailedTrialCount</td>
<td>3</td>
</tr>
</tbody>
</table>

### Objective

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>maximize</td>
</tr>
<tr>
<td>Goal</td>
<td>0.99</td>
</tr>
<tr>
<td>ObjectiveMetricName</td>
<td>Validation-accuracy</td>
</tr>
<tr>
<td>AdditionalMetricNames</td>
<td>accuracy</td>
</tr>
</tbody>
</table>
Demo: Configuring Search Space

Algorithm

**Algorithm Name**
- BayesianOptimization

Parameters

**Name** ~lr
- **Parameter Type**: double
- **Parameter Type**: int
- **Parameter Type**: categorical

- **Min**: 0.01
- **Min**: 2
- **Min**: 2

- **Max**: 0.03
- **Max**: 5
- **Max**: 5

**Trial Spec**
- **Trial Spec**: mnist-trial-template
Demo: Viewing Experiment Results
Demo: Viewing Trial Metrics