Intel® Cloud Optimization Modules for AWS*: XGBoost* on SageMaker*:

Custom Accelerated-AI Images for SageMaker

version 2023.Q3

SageMaker is a fully managed machine learning service on the AWS cloud. The motivation behind this platform is to make it easy to build robust machine learning pipelines on top of managed AWS cloud services. Unfortunately, the abstractions that lead to its simplicity make it quite difficult to customize.

This cheat sheet will explain how you can inject your custom training and inference code into a prebuilt SageMaker pipeline. Our main goal is to enable Intel AI Analytics Toolkit accelerated software in SageMaker pipelines.

The same methodology can be applied to other AI algorithms and workloads, with minor adjustments. If adapting to a deep learning workload, we recommend trying out some of intel's accelerated deep learning options like the Intel Extension for PyTorch and Intel Extension for Tensorflow.

SageMaker Pipeline* Setup

A detailed walkthrough of the module's implementation can be found in the GitHub Repo. Your pipeline.py file defines your sagemaker pipeline configuration.

Select your Training and Processing Instance:

```
def get_pipeline(processing_instance_type="ml.m5.xlarge", training_instance_type="ml.m5.xlarge")
```

Registered Model Inference Instance:

```
step_register = RegisterModel(inference_instances=["ml.m5.xlarge"])
```

Selecting XGBoost and daal4py Optimized Image:

```
xgb_train = Estimator(image_uri= <ECR URI>)
```

XGBoost* dmlc XGBoost v1.x+

Optimizations for training and prediction on CPU are upstreamed. Install the latest XGBoost with PyPi* or Anaconda* – newer versions have the most optimizations.

```
pip install xgboost
```

```
conda install xgboost -c conda-forge
```

Put data in XGBoost DMatrix:

```
DMatrix = xgb.DMatrix(
X_train.values, y_train.values)
```

Train XGBoost model:

```
model = xgb.train(params, Dmatrix,
num_boost_round=500)
```

Cheat Sheet | Docs | Medium Example

daal4py*

Speed up inference of the XGBoost model using daal4py, which is based on the Intel® oneAPI Data Analytics Library (oneDAL). Install the latest daal4py:

```
pip install daal4py
```

```
conda install daal4py –c conda-forge
```

Convert a model to daal4py format from XGBoost:

```
d4p_model =
d4p.get_gbt_model_from_xgboost(model)
```

For optimized inference:

```
prediction = (d4p.
gbt_classification_prediction(
nClasses, resultsToEvaluate)
.compute(data, model)
.probabilities[:,1]
```

GitHub Repo | Docs | Medium Example

Support Forums:

GitHub Repo | Intel DevHub Discord