May 2018

SageMaker Machine Learning Workshop

Lab: SageMaker In-Built Algorithms - XGboost

Table of Contents

1. Introduction

The following lab will teach you how to use SageMaker to quickly train a machine learning model using one of the SageMaker built-In algorithms (XGBoost specifically) and to host that trained model in a scalable faul-tolerant way to respond to client applications making low-latency inferencing requests.

1.1 Forming 2- or 3-Person Teams

First, if you are doing this lab as part of a large (>50 students) class, it is best if you can form a 3-person team and do the lab together as a group rather than as 3 separate individuals. It is also good if at least one of the members of the team are experts in using Jupyter notebooks. This will give an opportunity for discussion and troubleshooting within the group as the lab proceeds.   It also reduces that chances that the AWS account used by the customer won’t suddenly exceed its initial resource limits. Limits on the account can be lifted, of course, but those arrangements can take time to change and lifting the resource limits can increase the AWS costs for the account.

2. Setting Things Up

This lab assumes you have already been set up with an AWS user account with IAM permissions that allow SageMaker and related resources to be accessed and used. If not, contact your company's AWS administrator and have them create the user account and permissions for you.

2.1 Setting Up an S3 Bucket and Creating a SageMaker Notebook Instance

If you don't yet have an S3 Bucket ready for use by SageMaker, you will need to create one. Since S3 names have to be globally unique, and since SageMaker can automatically work with S3 buckets with names starting with "sagemaker", we recommend using a name with a pattern like "sagemaker-<initials><number>" where <initials> would be your 2- or 3-letter initials, and <number> would be a digit. For example "sagemaker-kls2".

And if you have not yet created a SageMaker Notebook Instance, you will also need to create one. We recommend using the default ml.t2.medium EC2 instance type for the notebook instance.

These tasks above can both be accomplished by following the AWS online documentation for SageMaker titled ***Getting Started*** (http://amzn.to/2nckMy4) and doing step 1.2 (click on step 1 and navigate to step 1.2) and step 2.

The result of the above steps is that you will have an appropriately-named S3 bucket to contain data and artifacts that will be generated and used by SageMaker, and you will have an available SageMaker notebook instance (Jupyter notebook) available for executing training.

3. Running the Lab's Notebook Within SageMaker

When a SageMaker notebook instance is created and opened, there will already be a large collection of sample notebooks installed. You can find them within the Jupyter notebook by navigating to the /sample-notebooks/ directory. In the following lab, we will be opening and running a particular sample notebook.

3.1 Lab: SageMaker In-Built Algorithms - XGBoost

This lab will make use of the xgboost_abalone sample notebook which demonstrates the use of Amazon SageMaker's implementation of the XGBoost Gradient-Boosted Tree algorithm to train and host a regression model. It uses the publicly-available Abalone dataset originally from the UCI data repository.

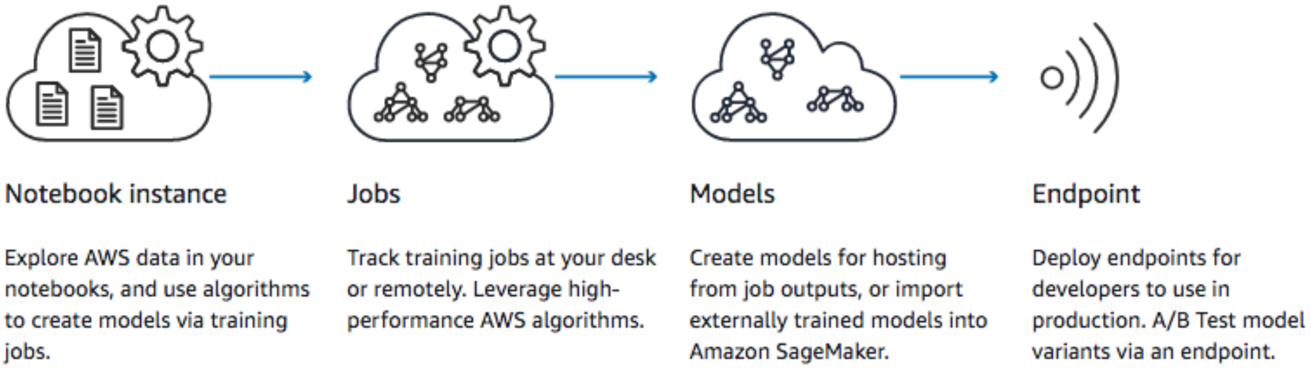
Once your SageMaker notebook instance is open and running, you can navigate to this lab's notebook in the following folder:

/sample-notebooks/introduction_to_amazon_algorithms/xgboost_abalone

To open and run the notebook, click on the **xgboost_abalone.ipynb** file. This will display the rendered text of the notebook and provide the executable cells.

At a high level, most SageMaker labs will follow a consistent sequence of activities along the Machine Learning *life cycle* - figure below. Typically these activities consist of the following steps:

1. Retrieving and converting a public sample dataset from some remote location into training data records and test data records.
2. Specifying a training job request detailing the location of the training data, the training algorithm to be used, and the type and number of EC2 instances to be used.
3. Submitting the training job request to the SageMaker Training Service to yield a trained model.
4. Deploying the trained model behind a SageMaker Endpoint to serve inferencing requests from client applications.



Notebook instance

Explore AWS data in your notebooks, and use algorithms to create models via training jobs.

Jobs

Track training jobs at your desk or remotely. Leverage high-performance AWS algorithms.

Models

Create models for hosting from job outputs, or import externally trained models into Amazon SageMaker.

Endpoint

Deploy endpoints for developers to use in production. A/B Test model variants via an endpoint.

Carefully read the explanation texts in the opened notebook to understand what kind of machine learning the notebook will perform, including an explanation of the dataset that will be used to train a model. Pay particular attention to any "Prerequisites & PreProcessing" section, such as replacing <value> placeholders with actual values specific to your account (e.g. your S3 bucket name). Then in sequence, run each cell in the notebook taking time to wait for long-running operations to report progress and final status to be displayed within the notebook.

Cells within the notebook will perform different aspects of the ML process. Initial ones may download data from a remote location while others will set things up to call the SageMaker Training service: e.g.

via a .fit() method call on an Amazon-provided algorithm, or instantiate a SageMaker client that dispatches a JSON-based training request to the training service. And once the model has been trained, other cells will ask SageMaker Hosting service to configure an endpoint and deploy the model with an endpoint that can be invoked by production client applications to make inferences on new data. The details of what the cells do are explained fully in the notebook.

While the SageMaker notebook is running, you can switch back to the browser tab containing the SageMaker web console and explore the assets that have been, and are being created, in SageMaker from notebook runs. Explore on the left-hand margin the training **Jobs**, trained **Models**, **Endpoint Configurations**, and **Endpoint** instances, and drill down into those assets if they've been created.

3.2 Exploring the Many Other Sample Notebooks in SageMaker

This particular lab's notebook was only one of many example notebooks that demonstrate how SageMaker can be used, the real power can be seen with the other comprehensive, real-world, and advanced sample notebooks present in your SageMaker notebook instance.

All of these notebooks can be found in the same **`sample-notebooks/`** directory and subdirectories. Each notebook when opened up individually within your SageMaker notebook instance will contain detailed explanations of that notebook. Before opening up each one to browse through what is available, there is also a listing and descriptions of the set of notebooks that can be viewed on github here: `http://bit.ly/2DDuHYB`. Remember that while Github also allows the same IPython notebook files to be viewed, you'll need to go back to your SageMaker notebook instance to open up and to run the notebook files.

Take time now to select one of the many sample notebooks that interests you, and then proceed to open it up in your SageMaker notebook instance and run it. You'll see how it is as easy and as consistent a process as this lab's example was earlier.

4. Cleaning Up

Since the resources you created in this lab are for practice only, failing to remove them can result in charges to the AWS account holding them. Most notebook examples will contain cells at the end that when executed will remove the artifacts created. However, you should be sure to use the SageMaker web console to delete any other endpoints, endpoint configurations, Models, Jobs, and even Notebook instances you may have left behind.