Training Machine Learning Models Using AWS SageMaker

Jorge Vásquez
SOFTWARE ENGINEER

@jorvasquez2301

Overview

Creating training jobs in SageMaker

Monitoring and analyzing training jobs

Automatic Hyperparameter Optimization

Creating tuning jobs in SageMaker

Monitoring and analyzing tuning jobs

Creating Training Jobs in SageMaker

To train a model in AWS SageMaker, you have to create a Training Job

AWS SageMaker Training Job

URL of the S3 bucket containing training data

Compute resources for training

URL of the S3 bucket where the job output will be stored

Elastic Container Registry for the training code

Monitoring and Analyzing Training Jobs

"If you can't measure it, you can't improve it."

Peter Drucker

Monitoring SageMaker with CloudWatch

AWS CloudWatch collects raw data and processes it into readable, near real-time metrics

Monitoring and Analyzing Training Jobs

A training job is an iterative process

It computes several metrics

Will the model generalize well?

Metrics are written to logs

SageMaker sends logs to AWS CloudWatch

You can view graphs of those metrics in CloudWatch

Training Job Instance Metrics

CPUUtilization

MemoryUtilization

GPUUtilization

GPUMemoryUtilization

DiskUtilization

train:accuracy

validation:accuracy

Metrics reported by the built-in Image Classification Algorithm

Defined at the moment of configuring the estimators

Metrics reported by the custom Tensorflow and MXNet algorithms

Training Jobs Logs in AWS CloudWatch

Anything an algorithm container sends to `stdout` or `stderr` is also sent to AWS CloudWatch Logs.

Training Jobs Logs in AWS CloudWatch

Log Group Name	Log Stream Name
/aws/sagemaker/TrainingJobs	[training-job-name]/algo-[instance-number-in-cluster]-[epoch_timestamp]

Demo

Creating and monitoring a training job for the built-in Image Classification algorithm, using the low-level AWS SDK for Python

Demo

Creating and monitoring a training job for the built-in Image Classification algorithm, using the high-level SageMaker Python library

Demo

Creating and monitoring a training job for the custom Tensorflow algorithm, using the high-level SageMaker Python library

Demo

Creating and monitoring a training job for the custom MXNet algorithm, using the high-level SageMaker Python library

Automatic Hyperparameter Optimization (HPO)

Selecting the right hyperparameter values for a machine learning model can be difficult. The right answer depends on the algorithm and the data.

Automatic Hyperparameter Optimization

Automatic HPO Finds the best version of a model by running many training jobs

Automatic Hyperparameter Optimization

It uses the algorithm and ranges of hyperparameters that you specify.

Chooses the hyperparameter values that result in a model that performs the best, as measured by a metric that you choose.

HPO is a supervised learning problem.
Given a set of input features (the hyperparameters), hyperparameter tuning optimizes a model for the metric that you choose.

Defining Objective Metrics

When using built-in algorithms

You don't need to define metrics

Metrics are sent automatically to hyperparameter tuning

You do need to choose the objective metric for the tuning job

When using custom algorithms

Your algorithm has to emit at least one metric by writing evaluation data to `stderr` or `stdout`

You can define up to 20 metrics for the tuning job to monitor

You choose one of those metrics to be the objective metric

You define metrics by specifying a name and a regular expression

Tunable Image Classification Hyperparameters

mini_batch_size

learning_rate

optimizer

Tunable Image Classification Hyperparameters

beta_1

beta_2

eps

gamma

momentum

weight_decay

Demo

Creating and monitoring a tuning job for the built-in Image Classification algorithm, using the low-level AWS SDK for Python

Demo

Creating and monitoring a tuning job for the built-in Image Classification algorithm, using the high-level SageMaker Python library

Demo

Creating and monitoring a tuning job for the custom Tensorflow algorithm, using the high-level SageMaker Python library

Demo

Creating and monitoring a tuning job for the custom MXNet algorithm, using the high-level SageMaker Python library

Summary

Creating training/tuning jobs

- Built-in Image Classification
- Tensorflow
- Apache MXNet

Monitoring and analyzing training/tuning jobs metrics and logs is easy with AWS Cloudwatch

Automatic HPO makes it easier to find the best hyperparameters combination for a given model