AWS SageMaker

Content Prepared By: Chandra Lingam, Cotton Cola Designs LLC

amazon web services

AWS Machine Learning Services

Service	Purpose
AWS Machine Learning	Easy to use Cloud Based ML service Perfect for beginners Linear Model - Regression, Binary Classification, Multinomial Classification
AWS SageMaker	Managed Jupyter Notebook environment Distributed Training Scalable Real Time Prediction Wide choice of ML Algorithms or bring your own
Application Services	Pre-trained Services Vision, Conversational Chatbots, Language Services

AWS SageMaker - Build

Managed Jupyter Notebook Environment

Use for Data Preparation

Pre-installed with popular ML Algorithms

Pre-configured to run TensorFlow and Apache MxNet

Bring-Your-Own Algorithm

AWS SageMaker - Train

SageMaker managed Training infrastructure

Distributed Training - Scale to Petabyte

Automatic Model Tuning

AWS SageMaker - Deploy

Deploy for Realtime predictions

Autoscaling Cluster across multiple availability zones

High performance and High Availability

Instance Families, Types and Pricing

Instance Families

Family	Strength
Standard	Balance of CPU and Memory
Compute Optimized	Large number of CPU Cores
Accelerated	Large number of GPU Cores

Pricing

Based on instance type, Storage, Data Transfer

Built-in Algorithms

Variety of Built-in Algorithms

- XGBoost (Competition winner!)
- Linear Learners
- Factorization Machines
- K-Means
- PCA
- And more

SageMaker Data Formats

Training Data Format

- CSV
- Protobuf RecordIO
- Algorithm specific formats
- Data needs to be stored in S3

Inference Format (for prediction from client application)

- CSV
- JSON
- Protobuf RecordIO

ML Terminology

Training Data – Used for training a model

Validation Data – Used for verifying training accuracy and for optimizing parameters

Test Data – Used for verifying accuracy of a built-up model (last step)

Data needs to be stored in S3

Algorithms Overview

Algorithm	Description
Linear Models	+ Simple + Performs surprisingly well for a variety of problems - Single equation trying to capture interaction of all variables - Categorical data needs to be encoded using one hot encoding
Decision Tree	+ Can Handle Complex non-linear relationship + Easily handles categorical data, missing data - Prone to overfitting - Poor predictive accuracy
Ensemble Methods	+ Combines multiple simple decision trees + Addresses Decision Tree overfitting problem + Much better predictive performance - More complex

Must Watch Videos

Gradient Boosting Machine Learning by Trevor Hastie

Learning Decision Tree by Alexander Ihler

Ensembles (Bagging) by Alexander Ihler

Demo 1 : Create S3 Bucket for SageMaker

- Create dedicated S3 bucket for the course
  - Data Store for training models
  - Model Artifact storage
  - Bucket Name: *<prefix>-ml-sagemaker*
- Sign-in with *my_admin* account

IAM Account Sign-in Link

https://<AccountId>.signin.aws.amazon.com/console
https://<Alias>.signin.aws.amazon.com/console

Demo 2 : Launch a Notebook Instance

Launch Notebook Instance – use *my_admin* account

- Define Permissions
- Select Instance Type
- Launch Notebook instance

Use ml_user account from this point onwards!

- Starter Samples
- Storing your custom code
  - Create a folder *SageMakerCourse*

Demo 3: Data Setup

- Download the following files from resources for this lecture: *XGBoostExamples.zip, extract_zip_file_content.ipynb*
- Upload the files to *SageMakerCourse* folder on the notebook instance
- Open *extract_zip_file_content.ipynb* to unzip the contents
- SageMakerSteps.xlsx – Contains file naming convention used

Demo 4: Data Formats and Interacting with S3

- Create sample file in CSV, RecordIO Formats
- Upload Files to S3
- Download Files from S3

Notebook: DataFormats\data_format_exploration.ipynb

Demo 5: XGBoost Regression

- Install XGBoost on Notebook Instance
- Regression Examples
    - Linear Model
    - Quadratic Model

Notebook: LinearAndQuadraticFunctionRegression\...

Demo 6: Kaggle Bike Sharing

- When used with AWS Machine Learning, RMSLE score was around 0.9
- Let's run the same example with XGBoost!

Notebook Folder: BikeSharingRegression

Demo 7: Kaggle Bike Sharing

- Optimization – Adding relevant features

Demo 8: Kaggle Bike Sharing

- Response variable as log1p

Demo 9: Train XGBoost Model on SageMaker

- SageMaker SDK Overview
- Prepare dataset for training, validation
- Train model
- Verify performance
- Deploy for Real-time prediction
- Query end point

Notebook: xgboost_cloud_training_template.ipynb

xgboost_cloud_prediction_template.ipynb

SageMaker Training Steps

1. Store data files in S3
2. Specify algorithm and hyper parameters
3. Configure and run the training job
4. Deploy the trained model

S3 Data Source Configuration

Attribute	Values/Purpose
S3DataDistributionType	FullyReplicated – entire dataset is replicated on each training instance ShardedByS3Key – Subset of data is replicated on each training instance. If dataset is split across multiple S3 objects, then SageMaker will distribute equal number of S3 objects to each training node.
S3DataType	ManifestFile – S3Uri points to a file that in-turn contains a list of files to be used for training S3Prefix – S3Uri points to a prefix. SageMaker uses all the objects with the specified prefix
S3Uri	Identifies a Key name prefix or a manifest file

Demo 10 – Invoke Prediction from outside

BikeSharingRegression

invoke_sagemaker_runtime_from_outside.ipynb

Pre-requisites:

1. Anaconda Python
2. Boto3 Library
3. SageMaker Library
4. ml_user_predict account as specified in house keeping lecture

Hyper Parameter Tuning

n_estimators (in XGBRegressor) is same as **num_round** (in XGBoost and SageMaker documentation)

This parameter controls number of rounds of boosting i.e. total number of trees.

Make sure you use correct parameter depending on the library. *XGBRegressor silently ignores parameters it does not understand* ☹

Hyper Parameter Tuning

XGBoost Tuning Suggestions

SageMaker XGBoost Hyper Parameter Documentation

Demo 11: XGBoost Classification

- Iris Model (categorical response, multi-class)
- Diabetes Dataset Model (binary classification)
- Mushroom Classification (categorical variables and response)
- Requires encoding categorical data to numeric for training and prediction