Machine Learning on AWS with SageMaker

Joe Wilson
Solutions Architecture, AWS
Worldwide Public Sector
Agenda

• Introduction + Recap
• AI Services
• SageMaker Deep Dive
  • Build / train / tune
• SageMaker Deep Dive
  • Deploy
  • Prepare
• Q/A
The AWS ML Stack

Broader and most complete set of machine learning capabilities

<table>
<thead>
<tr>
<th>AI SERVICES</th>
<th>HEALTH AI</th>
<th>INDUSTRIAL AI</th>
<th>ANOMALY DETECTION</th>
<th>CODE AND DEVOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon HealtlhLake</td>
<td>Amazon Transcribe for Medical</td>
<td>Amazon Monitor for Equipment</td>
<td>Amazon Lookout for Vision</td>
<td>Amazon DevOps Guru</td>
</tr>
<tr>
<td>Amazon Transcribe for Medical</td>
<td>Amazon Panorama + Appliance</td>
<td>Amazon Lookout for Equipment</td>
<td>Amazon Lookout for Metrics</td>
<td>Amazon CodeGuru</td>
</tr>
<tr>
<td>Amazon Comprehend for Medical</td>
<td>Amazon Transcribe for Medical</td>
<td>Amazon Lookout for Equipment</td>
<td>Amazon Lookout for Metrics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VISION</th>
<th>SPEECH</th>
<th>TEXT</th>
<th>SEARCH</th>
<th>CHATBOTS</th>
<th>PERSONALIZATION</th>
<th>FORECASTING</th>
<th>FRAUD</th>
<th>CONTACT CENTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Rekognition</td>
<td>Amazon Polly</td>
<td>Amazon Transcribe</td>
<td>Amazon Comprehend</td>
<td>Amazon Translate</td>
<td>Amazon Kendra</td>
<td>Amazon Lex</td>
<td>Amazon Personalize</td>
<td>Amazon Forecat</td>
</tr>
<tr>
<td>Amazon Transcribe</td>
<td>Amazon Transcribe</td>
<td>Amazon Transcribe</td>
<td>Amazon Textract</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ML SERVICES</th>
<th>FRAMEWORKS &amp; INFRASTRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW: Amazon SageMaker</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Label data</td>
<td>mxnet</td>
</tr>
<tr>
<td>NEW: Aggregate &amp; prepare data</td>
<td>GLUON</td>
</tr>
<tr>
<td>NEW: Store &amp; share features</td>
<td>Keras</td>
</tr>
<tr>
<td>Auto ML</td>
<td>Deep Learning AMIs &amp; Containers</td>
</tr>
<tr>
<td>Spark/R</td>
<td>GPUs &amp; CPUs</td>
</tr>
<tr>
<td>NEW: Detect bias</td>
<td>Elastic Inference</td>
</tr>
<tr>
<td>Visualize in notebooks</td>
<td>Habana Gaudi</td>
</tr>
<tr>
<td>Pick algorithm</td>
<td>Trainium Inferentia</td>
</tr>
<tr>
<td>Train models</td>
<td>FPGA</td>
</tr>
<tr>
<td>Tune parameters</td>
<td>NEW: SageMaker JumpStart</td>
</tr>
<tr>
<td>NEW: Debug &amp; profile</td>
<td>NEW: Model management for edge devices</td>
</tr>
<tr>
<td>Deploy in production</td>
<td>Manage &amp; monitor</td>
</tr>
<tr>
<td>Manage &amp; monitor</td>
<td>NEW: CI/CD</td>
</tr>
<tr>
<td>NEW: CI/CD</td>
<td>Human review</td>
</tr>
</tbody>
</table>

© 2023, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Amazon SageMaker overview

**PREPARE**

SageMaker Ground Truth
Label training data for machine learning

**SageMaker Data Wrangler**
Aggregate and prepare data for machine learning

**SageMaker Processing**
Built-in Python, BYO R/Spark

**SageMaker Feature Store**
Store, update, retrieve, and share features

**SageMaker Clarify**
NEW
Detect bias and understand model predictions

**BUILD**

SageMaker Studio Notebooks
Jupyter notebooks with elastic compute and sharing

**Built-in and Bring your own Algorithms**
Dozens of optimized algorithms or bring your own

Local Mode
Test and prototype on your local machine

**SageMaker Autopilot**
Automatically create machine learning models with full visibility

**SageMaker JumpStart**
NEW
Pre-built solutions for common use cases

**TRAIN & TUNE**

Managed Training
Distributed infrastructure management

SageMaker Experiments
Capture, organize, and compare every step

Automatic Model Tuning
Hyperparameter optimization

**Distributed Training**
NEW
Training for large datasets and models

**SageMaker Debugger**
NEW
Debug and profile training runs

**Managed Spot Training**
Reduce training cost by 90%

**DEPLOY & MANAGE**

Managed Deployment
Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration
Simplify Kubernetes-based machine learning

**Multi-Model Endpoints**
Reduce cost by hosting multiple models per instance

SageMaker Model Monitor
Maintain accuracy of deployed models

SageMaker Edge Manager
NEW
Manage and monitor models on edge devices

SageMaker Pipelines
NEW
Workflow orchestration and automation

**SageMaker Studio**
Integrated development environment (IDE) for ML
Developing on Amazon SageMaker

The “build” part of build, train, deploy
Amazon SageMaker Studio
Amazon SageMaker Studio

You!
- Can enable IAM or single sign on
- Fast notebook start-up
- Consolidated experience

Your IDE
- Low-lift / automatic feature access
  - Autopilot
  - Experiments
  - Debugger
  - Model Monitoring
  - Versioning
  - Endpoint management

Your Compute
- Can create a shareable link with package dependences in single click
- Billing is per usage, not up time.
- Change environments per project

Remember! Features are backwards compatible, but differ in ease-of-use across instances vs Studio.
1-Click Notebook Sharing & Instance Upgrade

Domain

- EC2 Instance
- EC2 Instance
- EC2 Instance

Studio defaults to 1:1 user to instance mapping, and you can share content on these via snapshots.
Amazon SageMaker brings elastic, dedicated compute

Person  Project  Dataset  Step in your ML lifecycle

Allowing you to scale and manage each independently
Training on Amazon SageMaker

The “train” part of build, train, deploy
Many Ways to Train Models on Amazon SageMaker

1. Built-in Algorithms
2. Script Mode
3. Docker container
4. Locally*
5. In a pipeline
Train with your own XGBoost script

```python
# Open Source distributed script mode
from sagemaker.session import Session
from sagemaker.inputs import TrainingInput
from sagemaker.xgboost.estimator import XGBoost

est = XGBoost(entry_point='abalone.py',
               framework_version='1.2-1',  # Note: framework_version is mandatory
               hyperparameters=hyperparams,
               role=role,
               instance_count=2,
               instance_type=instance_type,
               output_path=output_path)

train_input = TrainingInput("s3:///{}/{}".format(bucket, prefix, "train"), content_type=content_type)
validation_input = TrainingInput("s3:///{}/{}".format(bucket, prefix, "validation"), content_type=content_type)
est.fit({'train': train_input, 'validation': validation_input})
```

2021-01-28 19:58:46 Starting - Starting the training job...
2021-01-28 19:59:10 Starting - Launching requested ML instancesProfilerReport-1611863926: InProgress

Bring custom Python packages in requirements.txt
Train with your own deep learning model

```python
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point='mnist.py',
                    role=role,
                    framework_version='1.8.0',
                    instance_count=2,
                    instance_type='ml.c4.xlarge',
                    hyperparameters={
                        'epochs': 6,
                        'backend': 'gloo'
                    })

inputs = sagemaker_session.upload_data(path='data', bucket=bucket, key_prefix=prefix)

estimator.fit({'training': inputs})
```
Train using a Docker File

```
FROM ubuntu:20.04
ARG DEBIAN_FRONTEND=noninteractive

# Don't prompt for tzdata on new versions of Ubuntu:
ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get -y update && apt-get install -y --no-install-recommends \\
    wget \\
    libcurl4-openssl-dev\n    libsodium-dev \n    r-base \n    r-base-dev \n    ca-certificates

RUN R -e "install.packages(c('mda', 'plumber'), repos='https://cloud.r-project.org')"

COPY mars.R /opt/ml/mars.R
COPY plumber.R /opt/ml/plumber.R

ENTRYPOINT ["/usr/bin/Rscript", "/opt/ml/mars.R", "--no-save"]
```

"AlgorithmSpecification": {
    "TrainingImage": '{}.dkr.ecr{}.amazonaws.com/sagemaker-rmars:latest'.format(account, region)
},
## Store, Search, Version, Track Model Artifacts by Default

### Search

<table>
<thead>
<tr>
<th>Property</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputDataConfig.DataSource.S3DataPath</td>
<td>Contains</td>
<td>sagemaker</td>
</tr>
</tbody>
</table>

### Results: Training jobs

<table>
<thead>
<tr>
<th>Name</th>
<th>Algorithm</th>
<th>Data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tuning-job-1-0863961fdd4a1258d-019-6c367d0</td>
<td>train: s3://sagemaker-us-east-1-865926247642/automl-churn-sdk-18-19-41-09/transformed-data/dppk/pb/train</td>
<td><img src="#" alt="Copy" /></td>
</tr>
</tbody>
</table>

### Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns.

- **Timestamp**
  - No old events at this moment. **Retry**
  - 2021-02-18T14:49:15.656-08:00 Docker entrypoint called with argument(s): train
  - 2021-02-18T14:49:15.656-08:00 Running default environment configuration script
  - [2021-02-18T14:49:16.656-08:00] [2021-02-18T20:49:16 INFO 139632547493632] Input channel configuration: ("validation": ["validation"]) (["text/csv", "TrainingInputMode": "Pipe", "S3DistributionType": "FullyReplicated", "RecordWrapperType": "None"], "train": ["Content-Type": "text/csv", "TrainingInputMode": "Pipe", "S3DistributionType": "FullyReplicated", "RecordWrapperType": "None"])
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:18 INFO 139632547493632] Number of GPUs being used: 0
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:18 INFO 139632547493632] Creating network in context cpu(0).
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=1, ...
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=1, speed=224.68175181008214 samples/sec
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=2, ...
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=2, speed=435.5759141042366 samples/sec
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=3, ...
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=3, speed=629.34978622412 samples/sec
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=4, ...
  - [2021-02-18T14:41:10.656-08:00] [2021-02-18T20:49:19 INFO 139632547493632] #progress notice: epoch=0, iterations=4, speed=816.327654798652 samples/sec

© 2023 Amazon Web Services, Inc. or its affiliates. All rights reserved
Reduce training costs with fully-managed spot instances

```python
from sagemaker.estimator import Estimator

est = Estimator(
    container,
    role,
    hyperparameters=hyperparameters,
    instance_count=1,
    instance_type=instance_type,
    volume_size=5,  # 5 GB
    output_path=output_path,
    sagemaker_session=sagemaker.Session(),
    use_spot_instances=True,
    max_run=3600,
    max_wait=7200,
    checkpoint_s3_uri=checkpoint_s3_uri)

est.fit('{{\textcomm': train_input}, job_name=job_name)
```

- Specify a max wait time
- SageMaker will default to giving you the lowest possible cost
- Store model checkpoints in S3 in case your job is interrupted for BYOM
- Many built-in algorithms automatically revert to a training job
- We have examples
- Save up to 90%

Amazon SageMaker price reductions: Up to 18% lower prices on ml.p3 and ml.p2 instances

by Urvashi Chowdhary | on 07 OCT 2020 | in Amazon SageMaker, Artificial Intelligence | Permalink | Comments | Share
Deploying on Amazon SageMaker

The “deploy” part of build, train, deploy
Deploy any open source model with Amazon SageMaker
Enabling autoscaling on your endpoint is easy

1. Use boto3’s SageMaker client to get the resource ID

```python
client = boto3.client('sagemaker')
response = client.describe_endpoint(
    EndpointName=endpoint_name)
variant_name = response['ProductionVariants'][0]['VariantName']
resource_id = 'endpoint/{}/variant/{}'.format(endpoint_name, variant_name)
```

2. Use boto3’s application autoscaling client to register a scalable target

```python
scaling_client = boto3.client('application-autoscaling')
response = scaling_client.register_scalable_target(
    ServiceNamespace='sagemaker',
    ResourceId=resource_id,
    ScalableDimension='sagemaker:variant:DesiredInstanceCount',
    MinCapacity=min_capacity,
    MaxCapacity=max_capacity,
    RoleARN=role)
```
Bring your own pretrained models to host on SageMaker using script mode

```python
from sagemaker.tensorflow import TensorFlowModel

model = TensorFlowModel(model_data='s3://mybucket/model.tar.gz', role='MySageMakerRole')
predictor = model.deploy(initial_instance_count=1, instance_type='ml.c5.xlarge')
```

You can use the built-in inference script with the deep learning container, **or you can bring your own inference script**.

```python
pytorch_model = PyTorchModel(model_data='s3://my-bucket/my-path/model.tar.gz', role=role, entry_point='inference.py')
predictor = pytorch_model.deploy(instance_type='ml.c4.xlarge', initial_instance_count=1)
```
Deploy *any open source model* on SageMaker with Docker

```
ARG REGISTRY_URI
FROM ${REGISTRY_URI}/mxnet-inference:1.6.0-cpu-py3
RUN pip install autogluon
RUN pip install PrettyTable

# Defines inference.py as script entrypoint
ENV SAGEMAKER_PROGRAM inference.py
```
More features in SageMaker Studio
Amazon SageMaker Autopilot

Automatic model creation with full visibility & control

Quick to start
Provide your data in a tabular form & specify target prediction

Automatic model creation
Get ML models with feature engineering & model tuning automatically done

Visibility & control
Get notebooks for your models with source code

Recommendations & Optimization
Get a leaderboard & continue to improve your model
Use Amazon SageMaker Autopilot to create and review top performing regression and classification models

<table>
<thead>
<tr>
<th>Trial name</th>
<th>Status</th>
<th>Start time</th>
<th>End time</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-sagemaker-tuning-job-1</td>
<td>Completed</td>
<td>9 hours ago</td>
<td></td>
<td>0.9206119775772095</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-2</td>
<td>Completed</td>
<td>9 hours ago</td>
<td></td>
<td>0.9202479720115662</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-3</td>
<td>Completed</td>
<td>7 hours ago</td>
<td></td>
<td>0.9200050234794617</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-4</td>
<td>Completed</td>
<td>7 hours ago</td>
<td></td>
<td>0.919519072059631</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-5</td>
<td>Completed</td>
<td>9 hours ago</td>
<td></td>
<td>0.9191550016403198</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-6</td>
<td>Completed</td>
<td>7 hours ago</td>
<td></td>
<td>0.9190340042114258</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-7</td>
<td>Completed</td>
<td>8 hours ago</td>
<td></td>
<td>0.9189119935035706</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-8</td>
<td>Completed</td>
<td>8 hours ago</td>
<td></td>
<td>0.9186699886457825</td>
</tr>
<tr>
<td>my-sagemaker-tuning-job-9</td>
<td>Completed</td>
<td>8 hours ago</td>
<td></td>
<td>0.9186699886457825</td>
</tr>
</tbody>
</table>
Amazon SageMaker Experiments

Organize, track, and compare training experiments

- Tracking at scale
- Custom organization
- Visualization
- Metrics and logging
- Fast Iteration

Track parameters & metrics across experiments & users
Organize experiments by teams, goals, & hypotheses
Easily visualize experiments and compare
Log custom metrics using the Python SDK & APIs
Quickly go back & forth & maintain high-quality
Use Amazon SageMaker Experiments to track and manage thousands of experiments
Amazon SageMaker Debugger

Analysis and debugging, explainability, and alert generation

- **Relevant data capture**: Data is automatically captured for analysis.
- **Data analysis & debugging**: Analyze & debug data with no code changes.
- **Automatic error detection**: Errors are automatically detected based on rules.
- **Improved productivity with alerts**: Take corrective action based on alerts.
- **Visual analysis and debugging**: Visually analyze & debug from SageMaker Studio.
Amazon SageMaker is fully managed

One click model deployment

- Auto-scaling
- Low latency and high throughput
- Bring your own model
- Python SDK
- Deploy multiple models on an endpoint
Amazon SageMaker Model Monitor

Continuous monitoring of models in production

**Automatic data collection**
- Data is automatically collected from your endpoints

**Continuous Monitoring**
- Define a monitoring schedule and detect changes in quality against a pre-defined baseline

**Flexibility with rules**
- Use built-in rules to detect data drift or write your own rules for custom analysis

**Visual data analysis**
- See monitoring results, data statistics, and violation reports in SageMaker Studio

**CloudWatch Integration**
- Automate corrective actions based on Amazon CloudWatch alerts

© 2021, Amazon Web Services, Inc. or its Affiliates.
Use Amazon SageMaker Model Monitor to identify model drift and take action