Sagemaker for Production?

Jonas Laake
Infracoder Meetup 01.2020, Hamburg
What is AWS SageMaker

Definition AWS:

„Amazon SageMaker is a fully-managed service that enables data scientists and developers to quickly and easily build, train, and deploy machine learning models at any scale. Amazon SageMaker includes modules that can be used together or independently to build, train, and deploy your machine learning models. “
Functionality

› Notebooks
  › Jupyter notebook on specific instance type
  › Include Git Repos, Lifecycle configurations

› Training Jobs
  › Train a model using data from S3
  › Hyperparameter Tuning Jobs

› Inference
  › Inference Endpoints based on pretrained model artifacts located in S3
Levels of abstraction

**Usage**
- SageMaker Web UI
  - Click your Training jobs/ Endpoints
- High-level Python SDK
  - Commands like deploy(), fit() to be used in a SageMaker Notebook
- AWS SDK
  - Control the SageMaker API programmatically (cmp. boto3)

**Algorithms**
Build-in algorithms
- Provided by AWS

Own Container
- Build your own SageMaker compatible Docker container
High-level usage (Training job)

Code snipped from Udacity course boston housing example

```python
container = get_image_uri(session.boto_region_name, 'xgboost')
xgb = sagemaker.estimator.Estimator(container,  # The image name of the training container
    role,  # The IAM role to use (our current role in this case)
    train_instance_count=1,  # The number of instances to use for training
    train_instance_type='ml.m4.xlarge',  # The type of instance to use for training
    output_path='s3://{}/{}//output'.format(session.default_bucket(), prefix),  # Where to save the output (the model artifacts)
    sagemaker_session=session)  # The current SageMaker session

xgb.set_hyperparameters(max_depth=5,
    eta=0.2,
    gamma=4,
    min_child_weight=6,
    subsample=0.8,
    objective='reg:linear',
    early_stopping_rounds=10,
    num_round=200)
s3_input_train = sagemaker.s3_input(s3_data=train_location, content_type='csv')
s3_input_validation = sagemaker.s3_input(s3_data=val_location, content_type='csv')

taxgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

Usage for Production

Discussion SageMaker vs. alternatives
› Development: Notebooks vs. Jupyter Notebooks on EMR, ...
› Training: Training jobs vs. Fargate, EMR, ...
› Inference:
  › Real-time: Endpoints vs. Fargate, ECS, ...
  › Batch: Batch transform jobs vs. AWS Batch, ...

Project Conditions
› Multiple models called multiple times to build single result (many model calls)
› Batch Mode and Real-time API
› Sklearn Pipelines with new experimental features (Versioning)

https://www.ccisjm.com/teamdiscussion/
How to build your own SageMaker container

Local folder structure for scikitBring_your_own example container:

```
Container/
  Dockerfile
  build_and_push.sh
  decision_trees/
    nginx.conf
    predictor.py
    serve
    train
    wsgi.py
```
How to build your own SageMaker container

Training preparations:

1) Upload Docker image to Elastic Container Registry (ECR)

   ```
   docker build <image name>
   docker tag <image name> <repository name>
   docker push <repository name>
   ( <repository name> of form <account number>.dkr.ecr.<region>.amazonaws.com/<image name>:<tag> )
   ```

2) Load training data to S3

   ```
   aws s3 cp <from location> <to location>
   ```
How to build your own SageMaker container

In-container folder structure:

```
/opt/ml/
  input/  
    config/  
     hyperparameters.json  
     resourceConfig.json  
    data/  
     < channel_name: 'training'>  
      < input data >  
    model/  
     < model files >  
  output/  
   Failure/  
/opt/program/  
 < local_folder_name: 'decision_trees'>/
```

Dockerfile

```
COPY decision_trees /opt/program
WORKDIR /opt/program
```
How does AWS use custom containers

**Start training** (e.g. using Python in a SageMaker Notebook)

```python
import boto3
client = boto3.client('sagemaker')
client.create_training_job(
    TrainingJobName='DecisionTreeJob',
    HyperParameters={'tuning.objective_metric': 'balanced accuracy'},
    AlgorithmSpecification={'TrainingImage': '', ...},
    RoleArn='',
    InputDataConfig=[
        'ChannelName': 'training',
        'DataSource': {
            'S3DataSource': {'S3Uri': 's3://...', ...}
        }, ...]
}, OutputDataConfig={'S3OutputPath': 's3://...'},
ResourceConfig={'InstanceType': 'ml.m4.xlarge', ... },
StoppingCondition={'MaxRuntimeInSeconds': 3000}
)
```

What happens in the background:
- Start specified instance -> pull and run Docker image from ECR
- Load training data from S3 to container location
- Start **train** script which trains the Model -> writes pickled model artifact to output folder
- Load model artifact to S3 location in compressed format
How does AWS use custom containers

Create custom Sagemaker Endpoint for Inference

Create custom Sagemaker Endpoint for Inference

What happens in the background:

› Start specified instance and pull and run Docker image from ECR
› Load model artifact from S3 to container location
› Start serve script which starts the inference server
› (Shutdown endpoint when not longer in use)

What this tells us about SageMaker

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html
Amazon SageMaker Studio (currently only available in us-east-2)

› SageMaker Notebooks: switch hardware
› Sagemaker Processing: Run preprocessing, postprocessing, evaluation jobs
› SageMaker Experiments: Organize, track, compare Processing Jobs
› SageMaker Debugger: Save internal model state at periodic intervals
› SageMaker ModelMonitor: Detect quality deviations, receive alerts for deployed models
   › Infer schema based on training data
   › Automatically fetch statistics (for pre-build containers)
› SageMaker Autopilot: Automatic preprocessing, algorithm selection, model tuning, ...
   › Generates python code
   › Automatic hardware configuration
What I did not told you

› Everything is running on a dedicated instance (choose wisely)
› SageMaker heavily interacts with Cloudwatch (hyperparameter tuning, ...)
› Clustermode is possible for training as well as for inference
› You can chain your containers (preprocessing -> inference -> postprocessing)
› There is not only plain python available for Notebooks
  › R
  › Spark
  › MXNet
  › Tensorflow
  › PyTorch
Conclusion

› Fast way of getting started
› Tons of supplementary material (hard to get an easy overview)
› Suitable way of training models
› PoC Endpoints available
› Rapid development of new features for SageMaker
Discussion

› In which cases might SageMaker already be sufficient for use in real products?
› What else would be necessary?
› …?
Vielen Dank

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe