Benefits of using machine learning in cloud

1. Size of dataset
2. Clustered training
3. Hyperparameter tuning
4. Training / Serving Skew
5. De-coupling model from client
6. Autoscaling prediction

Source: https://www.udemy.com/from-0-to-1-tensorflow-for-deep-learning/learn/v4/overview
Amazon SageMaker

A **fully managed service** that enables **data scientists** and **developers** to quickly and easily **build** machine-learning based models **into production** smart applications.

- Notebook Instances
- 1P Algorithms
- ML Training Service
- ML Hosting Service

Notebook Instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn More

Notebook instance settings

Notebook instance name
tensorflow-demo

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type
mlt2.medium

IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.

AmazonSageMaker-ExecutionRole-20180105T211996

VPC - optional
Notebook Instances will have internet access independent of your VPC setting.

No VPC

Encryption key - optional
An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.
**Notebook Instance**  

**Get a jupyter notebook**

---

**TensorFlow BYOM: Train locally and deploy on SageMaker.**

1. **Introduction**
2. **Prerequisites and Preprocessing**
   - A. Permissions and environment variables
   - B. Model definitions
   - C. Data Setup
3. **Training the network locally**
4. **Set up hosting for the model**
   - A. Export from TensorFlow
   - B. Import model into SageMaker
   - C. Create endpoint
5. **Validate the endpoint for use**

**Note:** Compare this with the tensorflow bring your own model example

**Introduction**

This notebook can be compared to Iris classification example notebook in terms of its functionality. We will do the same classification task, but we will train the same network locally in the box from where this notebook is being run. We then setup a real-time hosted endpoint in SageMaker.

Consider the following model definition for IRIS classification. This mode uses the tensorflow.estimator.DNNClassifier which is a pre-defined estimator module for its model definition. The model definition is the same as the one used in the Iris classification example notebook.

**Prerequisites and Preprocessing**

**Permissions and environment variables**

Here we set up the linkage and authentication to AWS services. In this notebook we only need the roles used to give learning and hosting access to your data. The SageMaker SDK will use IAM default roles when needed. If the user executes --role, it does not return a role with the appropriate permissions, you'll...
Notebook Instance

**Pros**
- Easy access to data in cloud
- Web interface
- No installation of packages

**Cons**
- Cost
  - Free tier for 2 month
- Region limitation
  - Seoul is not supported
- Package version constraints
1P Algorithms

- **XGBoost, FM, and Linear** for classification and regression
- **Kmeans and PCA** for clustering and dimensionality reduction
- **Image classification with convolutional neural networks**
- **LDA and NTM for topic modeling, seq2seq for translation**

ML Training Service

Cost vs. Time

Source:
Distributed training

1. Scale to hundreds of GPU
2. Train outrageously large models
3. Customize models with few lines of codes

But...

Need to manage instance orchestration (Kubernetes or Mesos)

Need to manage fault tolerance

An example of managing kubernetes cluster
https://blog.openai.com/infrastructure-for-deep-learning/

Source: https://youtu.be/la_M6bCV91M?list=PLOU2XLYxmslKGc_NBolhTn2Qhraji53cv
Managed vs manual

Source: https://youtu.be/la_M6bCV91M?list=PLOU2XYmsIKGc_NB0iThTn20hraji53cv&t=1358
Managed Distributed Training With Flexibility

ML Training Service

- Matrix Factorization
- Regression
- Principal Component Analysis
- K-Means Clustering
- Gradient Boosted Trees
- And More!

Amazon provided Algorithms

- MXNet
- TensorFlow
- Spark

Bring Your Own Script (IM builds the Container)

Bring Your Own Algorithm (You build the Container)

CPU

GPU

HPO

AWS re:Invent © 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Source: https://www.slideshare.net/AmazonWebServices/new-launch-introducing-amazon-sagemaker-mcl365-reinvent-2017
ML Training Service

To use cloud services tf.estimator is required

Source: https://youtu.be/t64orpgS-E?list=PLQU2XLYxmsIKGc_NB0lhTn2Qhraji53cy&t=361
This document introduces Estimators—a high-level TensorFlow API that greatly simplifies machine learning programming. Estimators encapsulate the following actions:

- training
- evaluation
- prediction
- export for serving

You may either use the pre-made Estimators we provide or write your own custom Estimators. All Estimators—whether pre-made or custom—are classes based on the `tf.estimator.Estimator` class.

⚠️ Note: TensorFlow also includes a deprecated Estimator class at `tf.contrib.learn.Estimator`, which you should not use.

Advantages of Estimators

Estimators provide the following benefits:

- You can run Estimators-based models on a local host or on a distributed multi-server environment without changing your model. Furthermore, you can run Estimators-based models on CPUs, GPUs, or TPUs without recompiling your model.
Create a training job using the sagemaker.TensorFlow estimator

```python
from sagemaker.tensorflow import TensorFlow

mnist_estimator = TensorFlow(entry_point='mnist.py',
                              role=role,
                              training_steps=1000)
```

Preparing the TensorFlow training script

Your TensorFlow training script must be a **Python 2.7** source file. The current supported TensorFlow version is **1.4.0**. This training script **must contain** the following functions:

- `model_fn`: defines the model that will be trained.
- `train_input_fn`: preprocess and load training data.
- `eval_input_fn`: preprocess and load evaluation data.
- `serving_input_fn`: defines the features to be passed to the model during prediction.

**Version constraints ( Python 2.7, Tensorflow 1.4.0 )**

Source: [https://github.com/aws/sagemaker-python-sdk](https://github.com/aws/sagemaker-python-sdk)
ML Training Service

Training PyTorch is possible

Source: https://www.slideshare.net/AmazonWebServices/new-launch-introducing-amazon-sagemaker-mcl365-reinvent-2017
ML Training Service

Summary

- Easy to use distributed training
- Supporting various frameworks
  - Tensorflow
  - MxNet
  - PySpark
  - PyTorch* ...

[*]: BYO Docker Containers with SageMaker Estimators
Easy Model Deployment to Amazon SageMaker

ML Hosting Service

Versions of the same inference code saved in inference containers. Prod is the primary one, 50% of the traffic must be served there!

Inference Image

Amazon ECR

Model Artifacts

InstanceType: c3.xlarge
MinInstanceCount: 5
MaxInstanceCount: 20
ModelName: prod
VariantName: prodPrimary
VariantWeight: 50

One-Click!

mxnet

TensorFlow™

Amazon Provided Algorithms

AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Source: https://www.slideshare.net/AmazonWebServices/new-launch-introducing-amazon-sagemaker-mcl365-reinvent-2017
ML Hosting Service

Summary

- Easy to deploy
- Auto-Scaling Inference APIs
- A/B Testing
- Low Latency & High Throughput
- Bring Your Own Model
- Python SDK
<table>
<thead>
<tr>
<th>Feature</th>
<th>Amazon ML</th>
<th>Amazon SageMaker*</th>
<th>Azure ML Studio</th>
<th>Google Prediction API</th>
<th>Google ML Engine**</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Regression</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Clustering</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Anomaly detection</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Recommendation</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Ranking</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Algorithms</td>
<td>unknown</td>
<td>10 built-in + custom available</td>
<td>100+ algorithms and modules</td>
<td>unknown</td>
<td>TensorFlow-based</td>
</tr>
<tr>
<td>Frameworks</td>
<td>✗</td>
<td>TensorFlow, MXNet</td>
<td>✗</td>
<td>✗</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Graphical interface</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Automation level</td>
<td>high</td>
<td>medium</td>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>

*Both out-of-the-box features and possible custom-built features are marked as available in Amazon SageMaker.

**The features available in TensorFlow are respectively marked as available in Google ML Engine.

Conclusion

- Where is your data? - AWS vs GCP
- How big is your model? - Cluster vs Instance
- Try to use tf. estimator
- Which deep learning framework are you using?
  - Keep track model converters, model zoo
  - Training efficiency
  - Tutorials, examples, community
- Cost vs Time
Thank you