Amazon SageMaker Seminar

2019.11.28

Amazon SageMaker の基礎

Amazon Web Services Japan
Machine Learning Solutions Architect

Shoko Utsunomiya

自己紹介

宇都宮 聖子

- 機械学習ソリューションアーキテクト
  - 機械学習サービスを担当
  - 前々職は量子情報の研究者
  - 前職は自動車OEMで自動運転開発
  - 担当領域
    - 自動運転、AIヘルスケア、AI ゲーム、Startup
  - 好きなサービス
    - Amazon SageMaker

shokout

AWS 上で稼働する非常に多くの機械学習ワークロード

aws

アジェンダ

- 機械学習の課題に対する SageMaker のメリット
- SageMaker を利用した機械学習プロセス
- 機械学習の開発・学習・推論を効率化する基本機能
- まとめ

アジェンダ

- 機械学習の課題に対する SageMaker のメリット
- SageMaker を利用した機械学習プロセス
- 機械学習の開発・学習・推論を効率化する基本機能
- まとめ

機械学習における “Undifferentiated Heavy Lifting”

開発環境構築

- 必要なリソースの見積もりと購入の決断
- 開発チームで均一な開発環境構築
- フレームワークのインストール，バージョン管理

機械学習モデルの学習

- CPU/GPU など用途にあったハードウェア環境提供
- スケーラブルな分散学習構築と広帯域な通信環境

運用

- 推論環境の準備とモデルのホスティング
- 機械学習と異なるスキルセットが求められる

Photo by Victor Freitas on Unsplash

AWS の提供する機械学習スタック

AI SERVICES

App developers with little knowledge of ML

Vision


Amazon Rekognition Image


Amazon Rekognition Video

Speech


Amazon Textract


Amazon Polly


Amazon Transcribe

Language


Amazon Translate


Amazon Comprehend

Chatbots


Amazon Lex

Forecasting


Amazon Forecast

Recommendations


Amazon Personalize

ML SERVICES

ML developers and data scientists


Amazon SageMaker

Labeling

Ground Truth

Model development

Notebooks

Algorithms

Marketplace

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Training

Training

Optimization (Neo)

Hosting

Deployment

Hosting

ML FRAMEWORKS & INFRASTRUCTURE

ML researchers and academics

Frameworks

TensorFlow

mxnet

PYTORCH

Chainer

HOROVOD

Interfaces

GLUON

Keras

Infrastructure


Amazon EC2 P3 & P3DN


Amazon EC2 C5


FPGAs


AWS Greengrass


Amazon Elastic Inference


Amazon Inferentia

aws

AWS の提供する機械学習スタック

マネージドサービスを活用し ビジネスの価値にフォーカス

ML SERVICES

ML developers and data scientists

Amazon SageMaker

Labeling

Ground Truth

Model development

Notebooks

Algorithms

Marketplace

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Training

Training

Optimization (Neo)

Hosting

Deployment

Hosting

ML FRAMEWORKS & INFRASTRUCTURE

ML researchers and academics

Frameworks

TensorFlow

mxnet

PYTORCH

Chainer

HOROVOD

Interfaces

GLUON

Keras

Infrastructure

Amazon EC2 P3 & P3DN

Amazon EC2 C5

FPGAs

AWS Greengrass

Amazon Elastic Inference

Amazon Inferentia

aws

Amazon SageMaker とは

- 機械学習プロジェクトの課題を解決するためのマネージドサービスで、数分で開発環境を起動でき、学習、推論環境は柔軟にスケール
- 多数のAPIを提供しており、他サービスとの自由度の高い連携が可能
- 東京を含む18リージョンで提供 (+Bahrain)
- ほとんどのコンテナ，SDKはオープンソース

NEW!

aws

Amazon SageMaker とは

- 機械学習プロジェクトの課題を解決するためのマネージドサービスで、数分で開発環境を起動でき、学習、推論環境は柔軟にスケール
- 多数のAPIを提供しており、他サービスとの自由度の高い連携が可能
- 東京を含む18リージョンで提供
- ほとんどのコンテナ，SDKはオープンソース

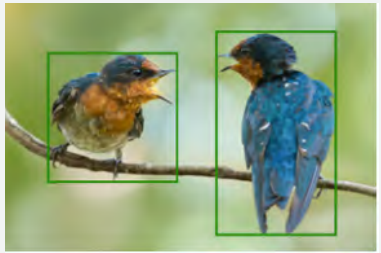
Amazon SageMaker Ground Truth

データにラベル (Ground Truth) を付与するアノテーション作業の支援サービス

- アノテーションにおける一般的なワークフローの管理ツール
- ラベルを付与するワーカーは，Amazon Mechanical Turk，外部ベンダ(AWS Marketplace)，自社のプライベートチーム の3つから選択
- 以下の4種類のタスク向けアノテーションツールも提供（カスタムも可能）

画像分類

物体検出



セマンティック セグメンテーション

文章分類

アノテーションの基本プロセス

1. アノテーション対象の
データをアップロード

Amazon S3

SageMaker
ユーザ

2. ラベリング
ジョブの作成

Amazon
SageMaker

3. タスクはワーカーに
自動で割り振られる

ワーカー

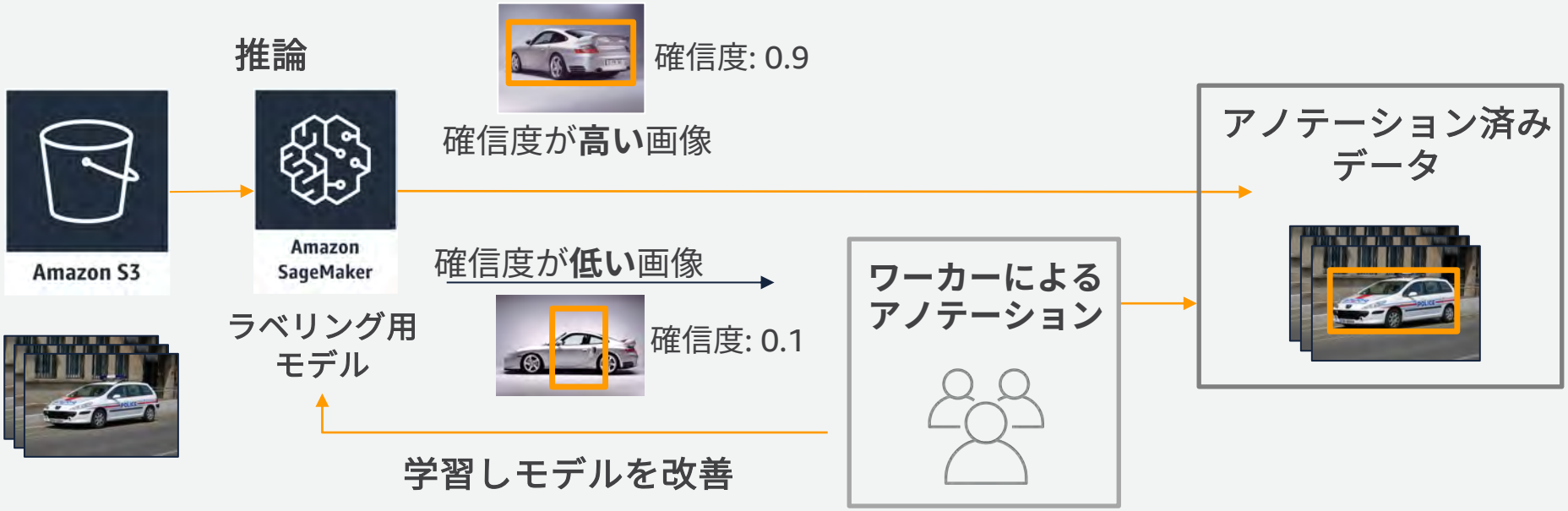
4. アノテーションツールで
ワーカーがアノテーションを行う

5. アノテーション
結果がS3集計される

複数人の結果を
マージできる

自動ラベリング (オプション機能)

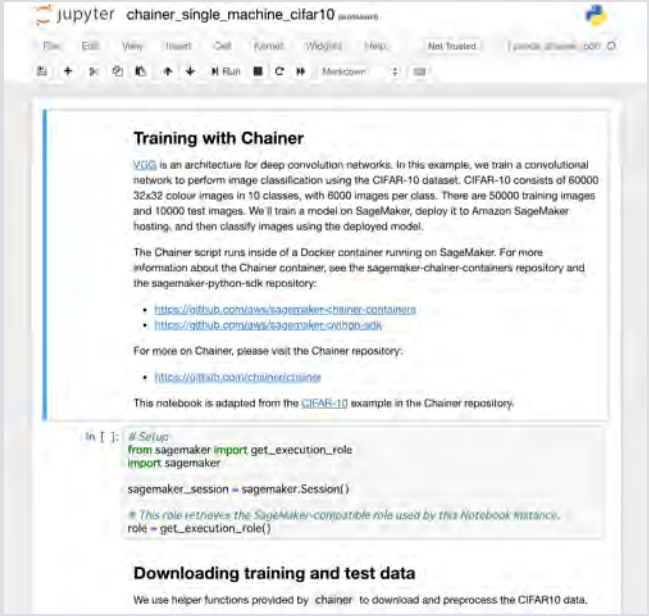
大規模データセット（5000データ程度）にラベリングする際、数割を
ワーカーでラベル付けし、残りを自動化することで、時間とコストを削減



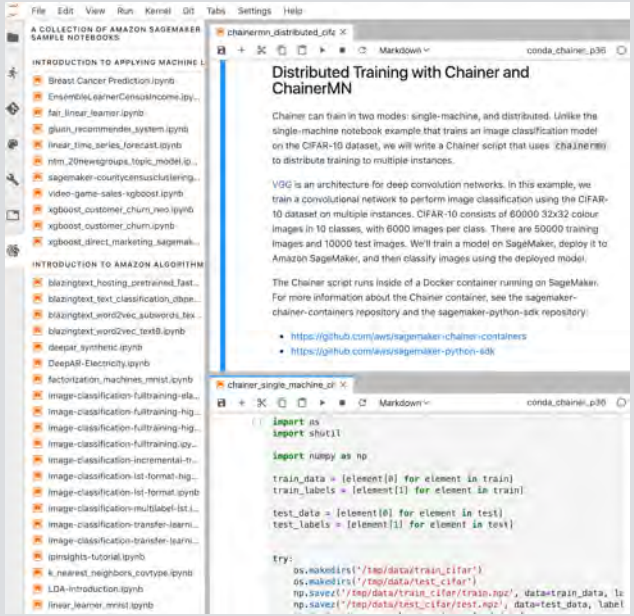
Amazon SageMaker Notebook instance

- SageMaker 上のワークフロー（環境・データのインポート，モデル定義，学習ジョブ，デプロイ，エンドポイント呼び出し）を SageMaker Python SDKで記述する

Jupyter Notebook



JupyterLab



学習: 分散学習や複数学習ジョブの同時実行

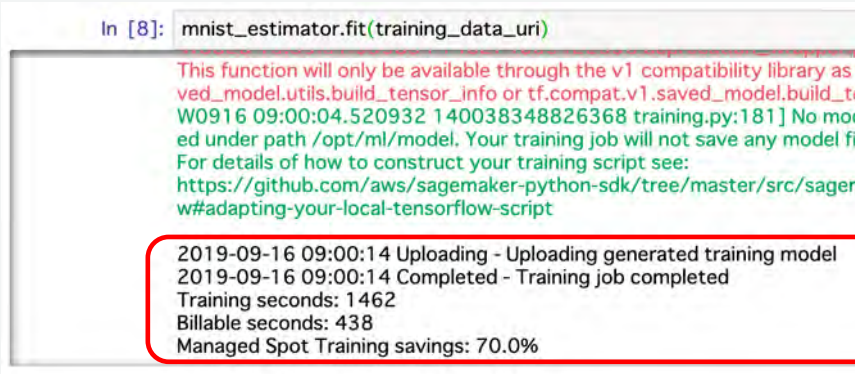
- APIを経由で学習用のインスタンスを起動可能で、**学習が完了すると自動で停止する**
- 高性能なインスタンスを手動で停止したりせずに済み、**簡単にコストを抑えることができる**
- 指定したインスタンス数で**分散学習環境が容易に構築できる**

SageMaker が Managed Spot Training に対応！

- これまでの学習コストを最大で90%削減
- すべてのフレームワーク，モデル，学習構成で利用可能
- Checkpointing により Spot instance が落ちても途中から学習を再開

NEW!

```
from sagemaker.tensorflow import TensorFlow

mnist_estimator = TensorFlow(entry_point='mnist.py',
                             role=role,
                             train_instance_count=2,
                             train_instance_type='ml.p2.2xlarge',
                             train_max_run = 5000,
                             train_use_spot_instances='True',
                             train_max_wait = 7200)
```



https://aws.amazon.com/jp/blogs/aws/managed-spot-training-save-up-to-90-on-your-amazon-sagemaker-training-jobs/

推論: API エンドポイントやバッチ推論

- 一般に作業負荷の大きい推論環境の構築を、**API1つで簡単に実現**
- 推論の負荷にあわせてGPUをアタッチできるElastic Inference
- バッチ推論を使えば、**必要なときだけエンドポイントを利用可能**
- エンドポイントは**オートスケール対応**
- **A/Bテスト**をサポート

オートスケールの設定やA/Bテスト用にリクエストの割り振りが可能

開発・学習・推論は個別に利用可能

例 1: 部署全員に対してマネージドノートブック環境を提供したい場合

管理不要のノートブックをホストする環境として SageMaker を利用

例 2: プロダクション環境がオンプレミスにすでにある場合

スケーラブルな学習環境としてのみ SageMaker を利用可能

例 3: オンプレミスに豊富な GPU クラスタを持っている場合

オンプレミスで学習済のモデルを AWS 上のプロダクション環境にデプロイ

アジェンダ

- 機械学習の課題に対する SageMaker のメリット
- SageMaker を利用した機械学習プロセス
- 機械学習の開発・学習・推論を効率化する基本機能
- まとめ

SageMaker の基本構成要素

Amazon S3　　Amazon SageMaker　　Amazon ECR

SageMaker
Python SDK

学習データ

学習スクリプト

DL / ML 実行環境

Tensor　PYTORCH　mxnet　Chainer　learn

aws

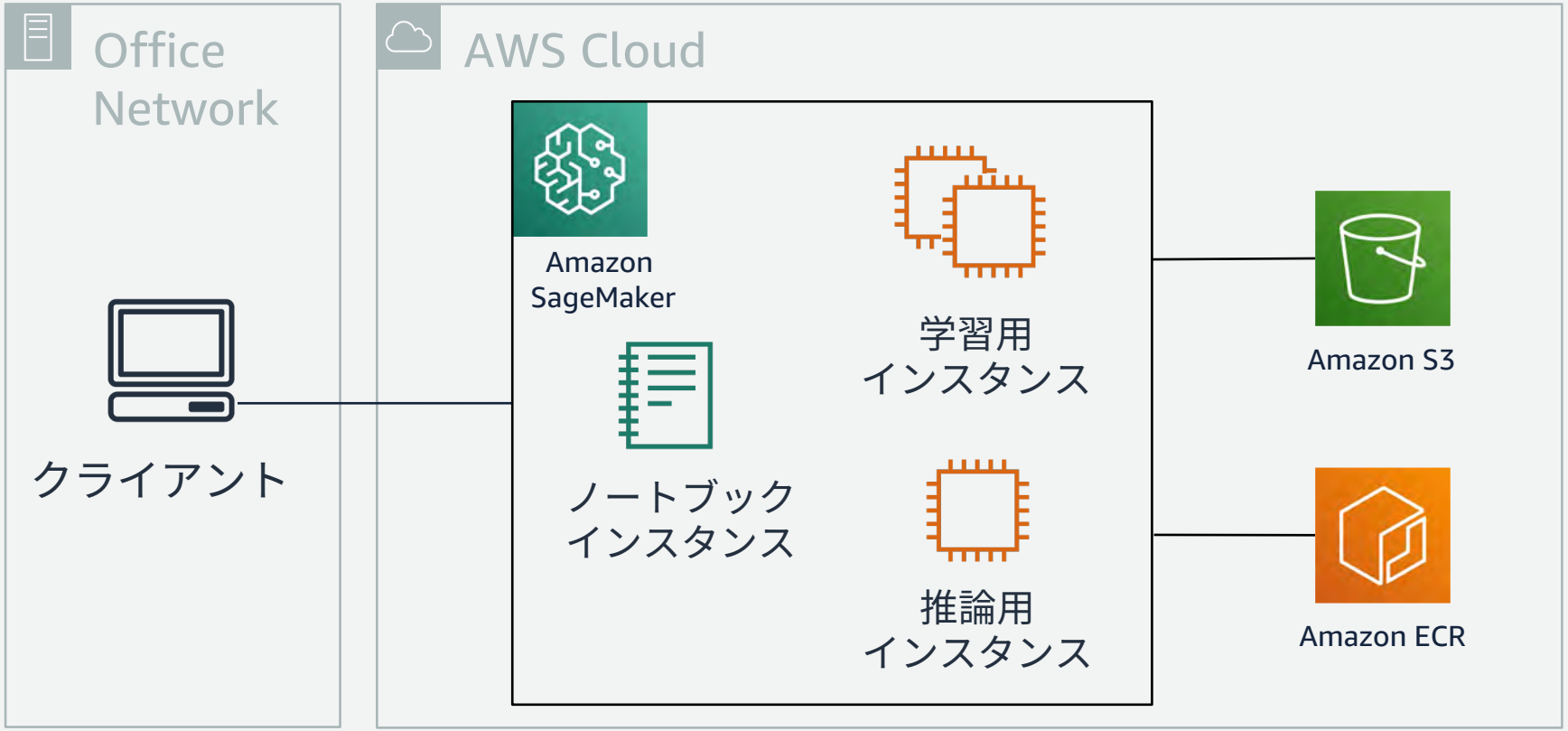
学習データの準備

- 学習データはオブジェクトストレージの Amazon S3におく
- 学習用のコードで読める形式であれば、学習データの形式は自由
- コンソールからでも、API 経由でもアップロード可能

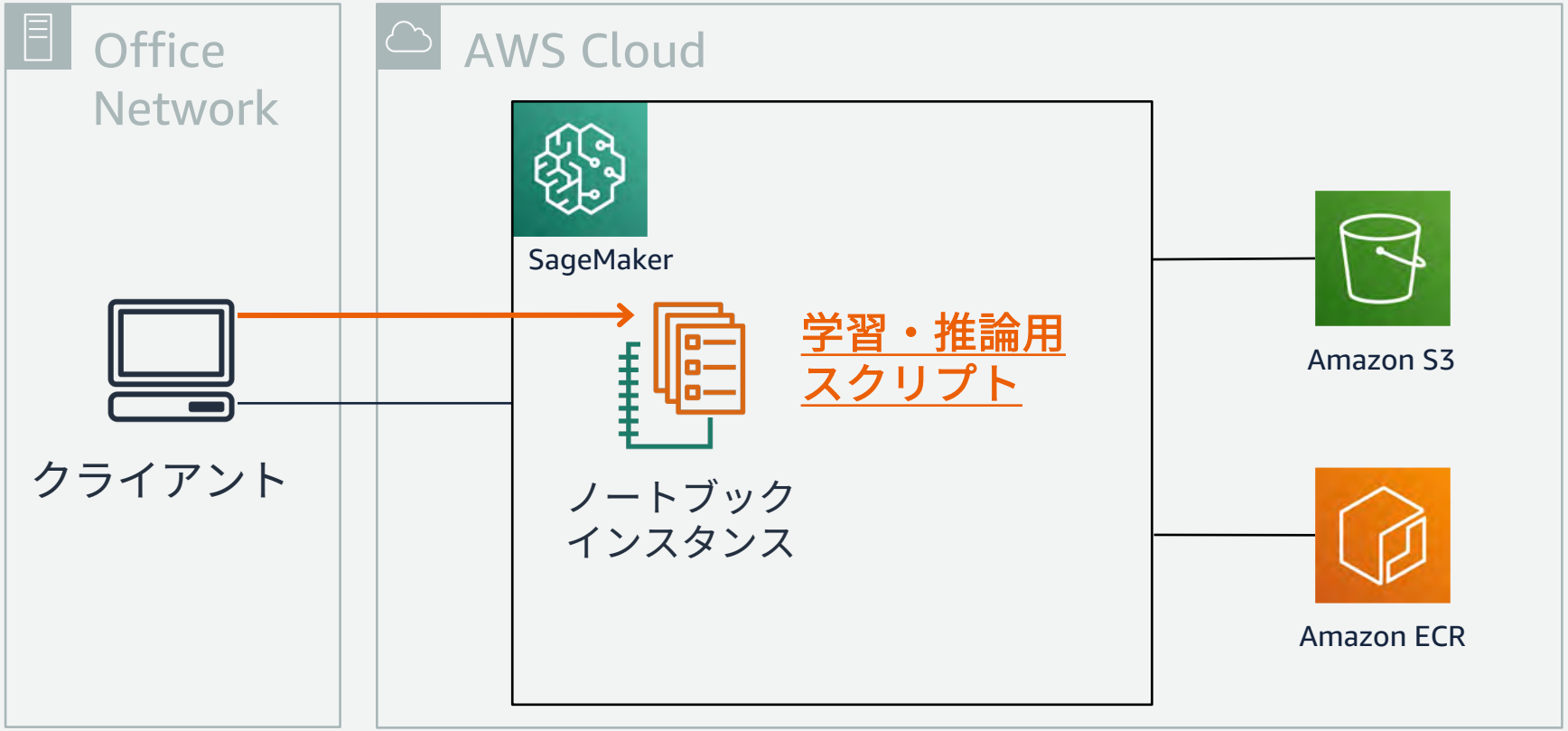
機械学習の実行環境をコンテナイメージで提供

- 実行環境をコンテナイメージとして Amazon ECR (Elastic Container Registry) におく
- ユーザは，自身が利用したい環境 (TensorFlow など) をコンテナイメージから選んで利用

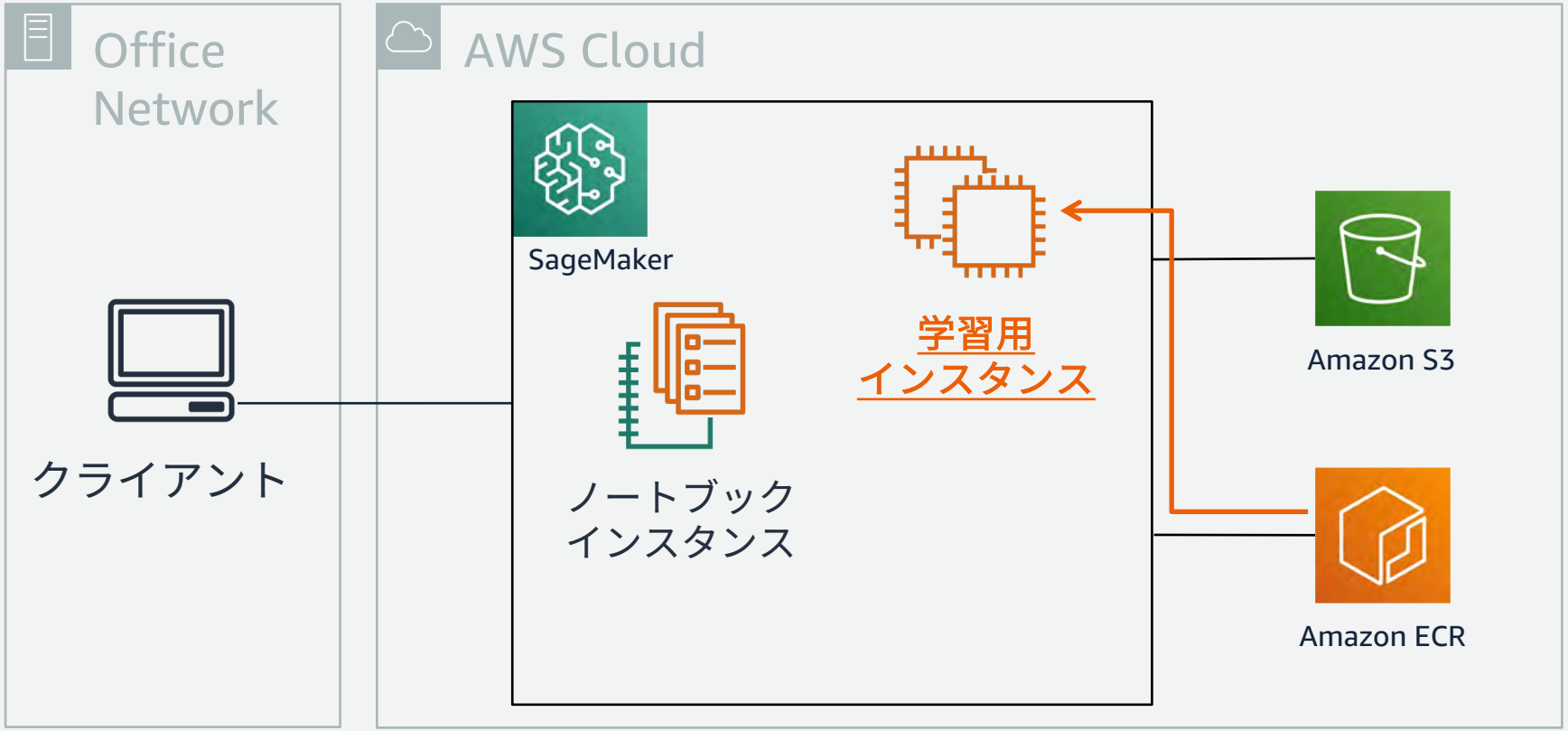
Amazon SageMaker のアーキテクチャ



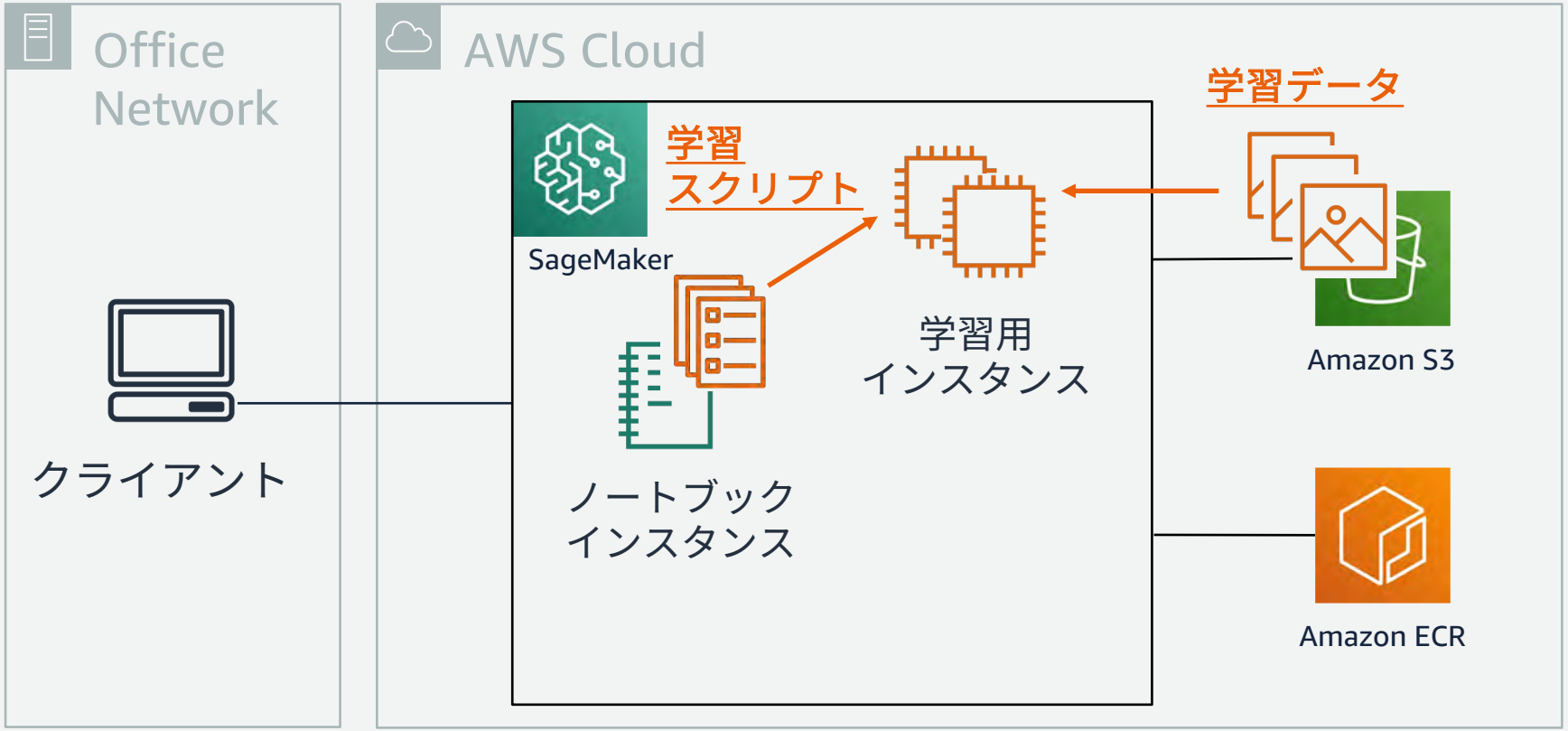
1. ノートブックインスタンスへコードを移行



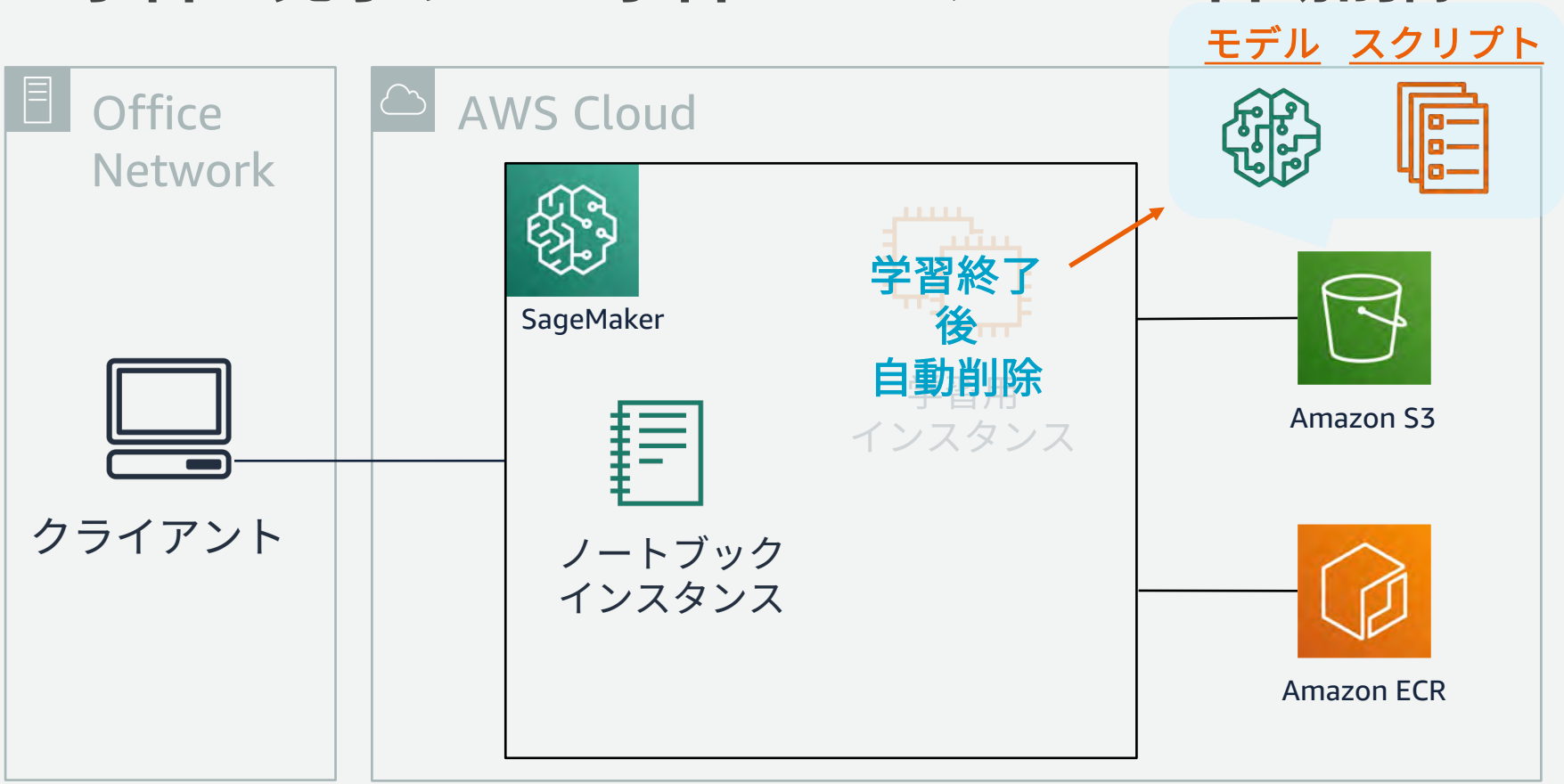
2. コンテナイメージから学習用インスタンスを起動



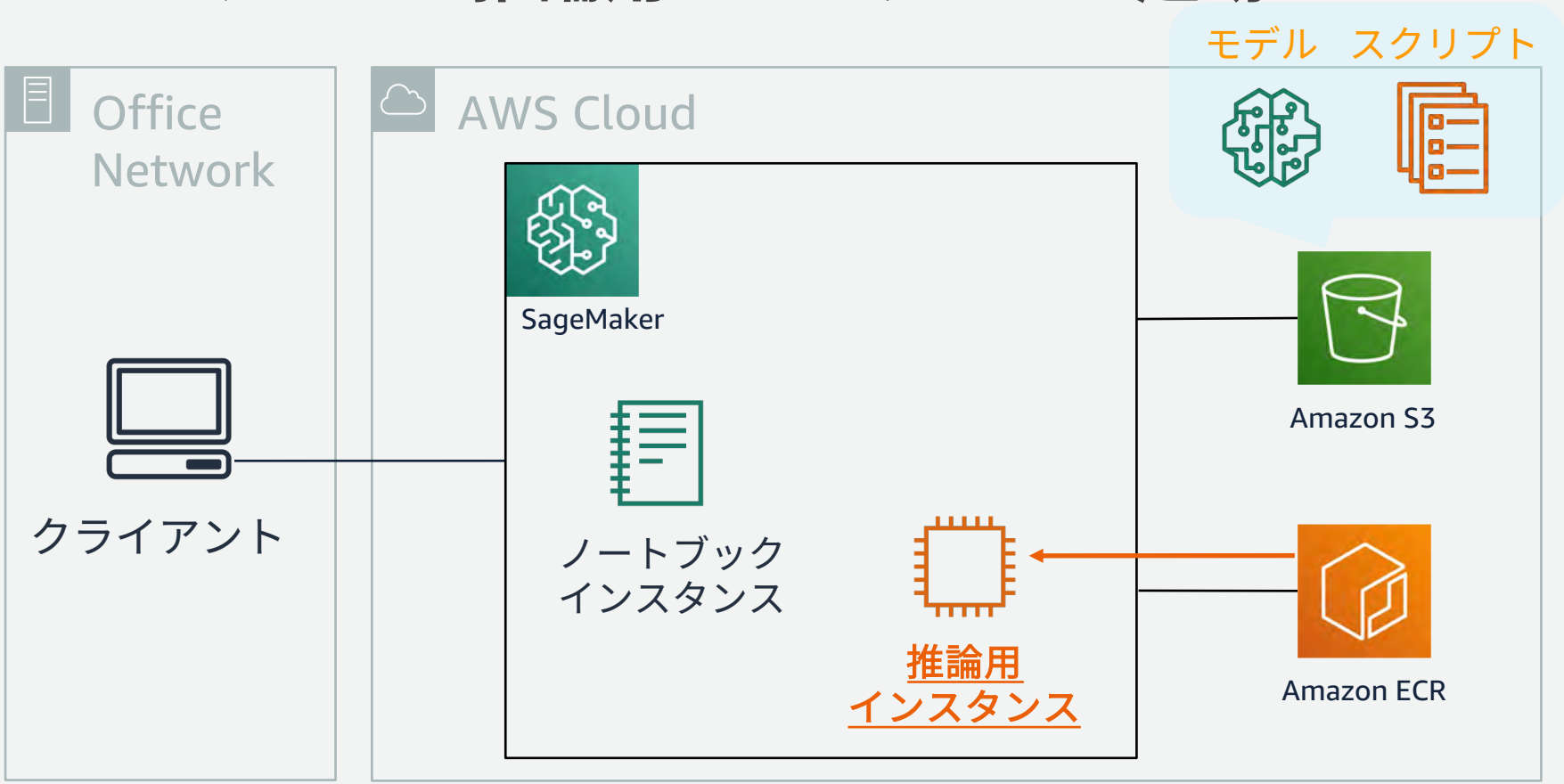
3. データとスクリプトを読み込み学習を実行



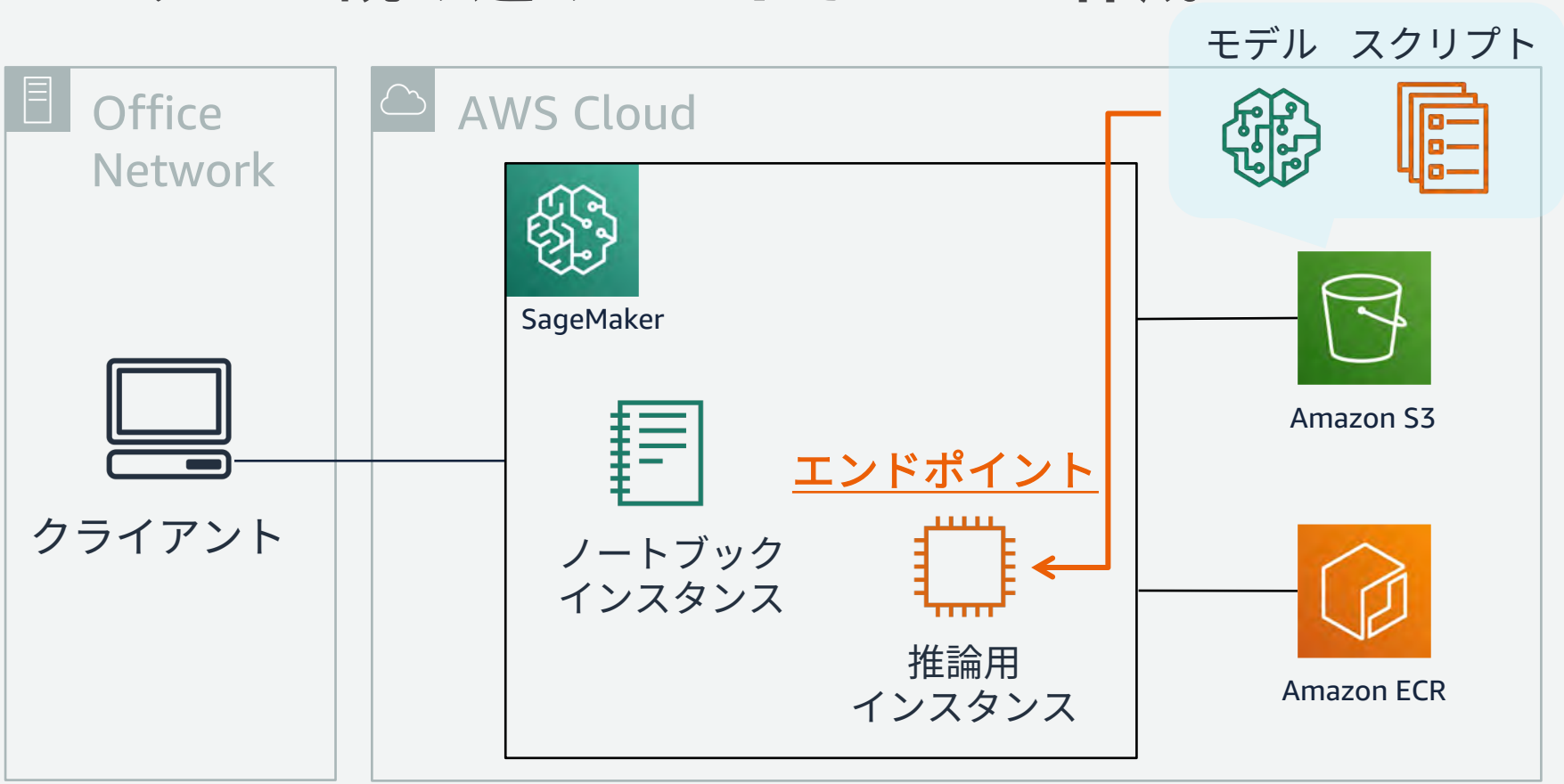
4. 学習が完了すると学習インスタンスは自動削除



5. コンテナから推論用インスタンスを起動



6. モデルを読み込みエンドポイント作成



SageMaker Python SDK による学習・推論の流れ

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py',
                          role='SageMakerRole',
                          train_instance_count=1,
                          train_instance_type='ml.p2.xlarge',
                          framework_version='1.12',
                          py_version='py3')

tf_estimator.fit('s3://bucket/path/to/training/data')

predictor = tf_estimator.deploy(initial_instance_count=1,
                                instance_type='ml.c5.xlarge',
                                endpoint_type='tensorflow-serving')

input = {'instances': [1.0, 2.0, 5.0]}
result = predictor.predict(input)
```

①Estimatorの作成

ジョブを実行するために，SageMaker用に準備された TensorFlow の Estimator クラスのオブジェクトを作成．学習スクリプト，学習インスタンスタイプ・数などを指定

②学習の実行

fit() を実行すると，指定したインスタンスが立ち上がり，用意された TF コンテナを読み込み，S3からデータをロードし，学習ジョブを実行する

③推論の実行

学習が終わったら，**deploy()** を実行すると，エンドポイントが作成される．推論用インスタンスと初期インスタンス数を指定．**predict()** で推論を実行

aws

SageMaker 学習コンテナと環境変数と S3

S3 location		環境変数	値
s3://<bucket>/<prefix>/train	→	SM_CHANNEL_TRAIN	/opt/ml/input/data/train
s3://<bucket>/<prefix>/test	→	SM_CHANNEL_TEST	/opt/ml/input/data/test
s3://<bucket>/<job_name>/model.tar.gz	←	SM_MODEL_DIR	/opt/ml/model
s3://<bucket>/<job_name>/output.tar.gz	←	SM_OUTPUT_DATA_DIR	/opt/ml/output/data

SageMaker に合わせた学習・推論コードの書き方

```python
class MLP(chainer.Chain):
    def __init__(self, n_units, n_out):
        super(MLP, self).__init__()
        ...

    def __call__(self, x):
        h1 = F.relu(self.l1(x))
        h2 = F.relu(self.l2(h1))
        return self.l3(h2)

if __name__=='__main__':

    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', type=int, default=50)
    ...
    parser.add_argument('--test', type=str, default=os.environ['SM_CHANNEL_TEST'])
    args, _ = parser.parse_known_args()

    train_data = np.load(os.path.join(args.train, 'train.npz'))['images']
    ...

    model = L.Classifier(MLP(1000, 10))
    optimizer = chainer.optimizers.Adam()
    optimizer.setup(model)
    ...
    trainer.run()
    serializers.save_npz(os.path.join(args.model_dir, 'model.npz'), model)

def model_fn(model_dir):
    model = L.Classifier(MLP(1000, 10))
    serializers.load_npz(os.path.join(model_dir, 'model.npz'), model)
    return model.predictor
```

ネットワーク定義はそのまま

①環境変数の引き渡し

ハイパーパラメータや入力データのパス等は，SageMaker 側で引数として引渡し，もしくはdefaultのままargparseで取り出す

②学習データのロード

S3 から学習実行時にコンテナへダウンロードされたデータのパスを指定

③学習済モデルのセーブ

学習済みのモデルを S3 に格納するためのパスを指定，S3 では tar.gz で保存される

④推論用のモデルロード

推論エンドポイントにおけるモデルのロード処理を，`model_fn()` 内に記述

aws

アジェンダ

- 機械学習の課題に対するSageMakerのメリット
- SageMakerを利用した機械学習プロセス
- 機械学習の開発・学習・推論を効率化する基本機能
- まとめ

開発・学習・推論を効率化する基本機能

Amazon S3　　Amazon SageMaker　　Amazon ECR

学習データ

学習スクリプト

DL / ML 実行環境

Tensor　PYTORCH　mxnet　Chainer　learn

aws

開発・学習・推論を効率化する基本機能

Amazon S3　　Amazon SageMaker　　Amazon ECR

学習データ

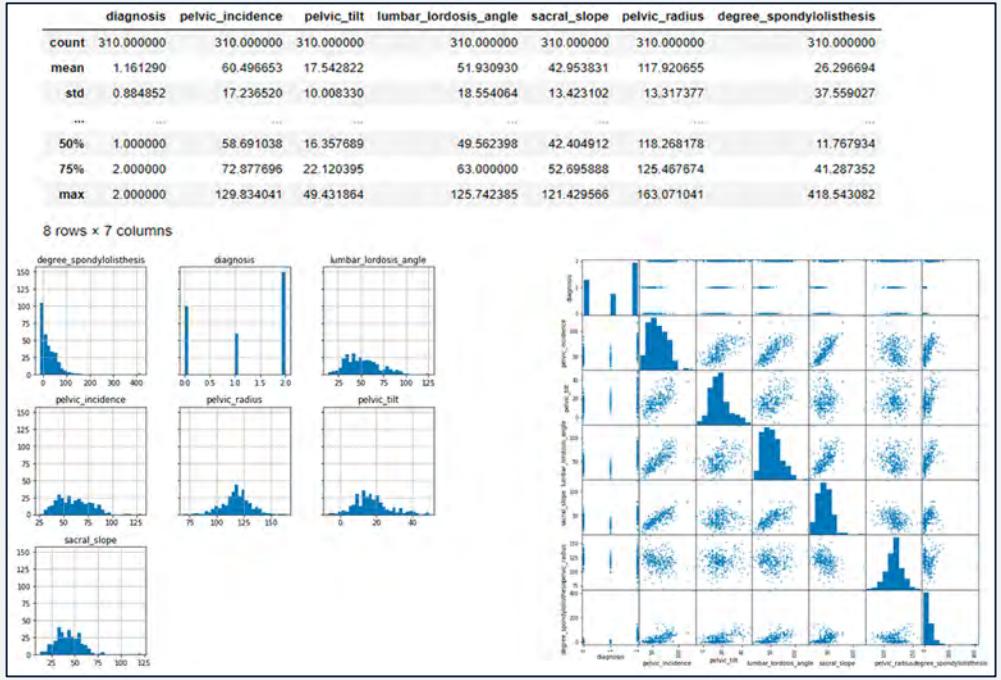
学習スクリプト

DL / ML
実行環境

- 学習データの前処理
- 大容量データの読み込み

ノートブックインスタンスでのデータ前処理

- EBSを16TBまで拡張可能
- 前処理に便利なNumpy, Pandas などのパッケージがプリインストール済み
- Jupyterで対話的に処理できる
- 前処理用に、EMRといった他のサービスを呼び出すことも可能



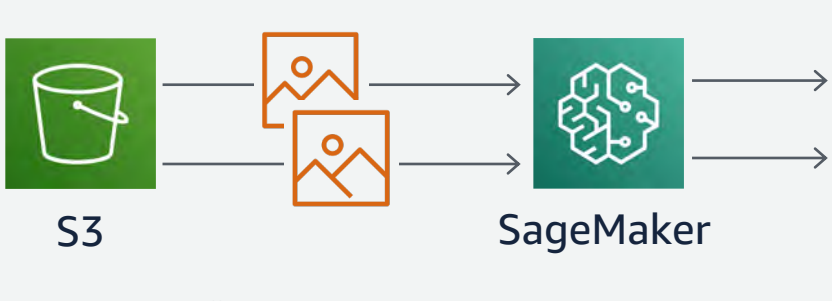
	diagnosis	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
count	310.000000	310.000000	310.000000	310.000000	310.000000	310.000000	310.000000
mean	1.161290	60.496653	17.542822	51.930930	42.953831	117.920655	26.296694
std	0.884852	17.236520	10.008330	18.554064	13.423102	13.317377	37.559027
...	...	...	...	...	...	...	...
50%	1.000000	58.691038	16.357689	49.562398	42.404912	118.268178	11.767934
75%	2.000000	72.877696	22.120395	63.000000	52.695888	125.467674	41.287352
max	2.000000	129.834041	49.431864	125.742385	121.429566	163.071041	418.543082

8 rows × 7 columns

S3からのファイル転送

FILE モード (Default): 全ファイルをダウンロードして学習

PIPE モード: ダウンロードしながら非同期に学習を実施



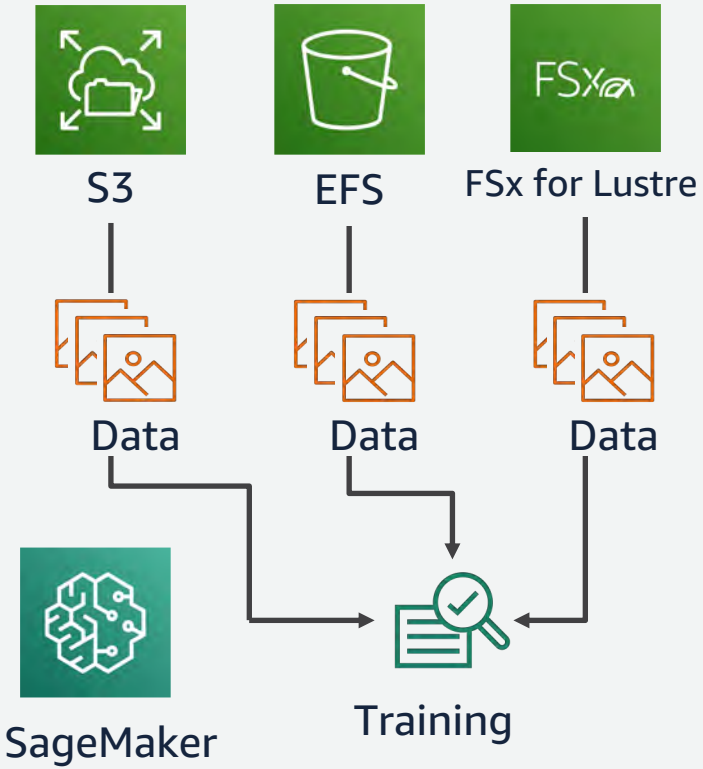
- ファイル転送のオーバーヘッドを軽減
- TensorFlow, MXNet のみ対応

aws

学習時のファイルシステムにEFSとFSx Lustreが対応

NEW!

- 学習用のファイルシステムとして，EFS と FSx Lustre を直接指定可能に
- 学習ジョブの開始時に S3から学習インスタンスのEBSへのデータ転送にかかっていたオーバーヘッドが削減される
- 同一ファイルを用いた反復の学習ジョブなどの場合では，高速キャッシュとしてファイルシステムを利用できる



https://aws.amazon.com/jp/blogs/machine-learning/speed-up-training-on-amazon-sagemaker-using-amazon-efs-or-amazon-fsx-for-lustre-file-systems/

開発・学習・推論を効率化する基本機能

Amazon S3　　Amazon SageMaker　　Amazon ECR

学習データ

学習スクリプト

DL / ML
実行環境

Tensor　PYTORCH　mxnet　Chainer　learn

- ビルトインアルゴリズム
- 自前でスクリプトを書く

aws

SageMaker ビルトインアルゴリズム ~ 機械学習モデル ~

モデル名	教師データ	アルゴリズム説明	利用用途の例
Linear Learner	あり	線形回帰	分類・回帰などの分析
XGBoost	あり	XGBoost, 勾配ブーストツリー (eXtreme Gradient Boosting)	分類・回帰などの分析
PCA	なし	主成分分析 (Principal Component Analysis)	次元削減
k-means	なし	K平均法	クラスタリング
k-NN	あり	K近傍法	クラスタリング
Factorization Machines	あり	行列分解	レコメンド，回帰，分類
Random Cut Forest	なし	robust random cut tree	時系列データの異常検知
LDA (Latent Dirichlet Allocation)	あり	生成的統計モデル	トピックモデル

※　LDAのオリジナルは教師なし

https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

aws

SageMaker ビルトインアルゴリズム ~ディープラーニング モデル~

	モデル名	教師データ	アルゴリズム	利用用途の例
画像処理	Image classification	あり	ResNet	画像の多値分類
	Object Detection	あり	SSD (Single Shot multibox Detector)	物体の画像内領域をバウンディングボックスで検出
	Semantic Segmentation	あり	FCN, PSP, DeepLabV3 (ResNet50, ResNet101)	ピクセル単位の画像内の物体領域検出
自然言語処理	seq2seq	あり	Deep LSTM	テキスト要約，音声認識
	Neural Topic Model	なし	NTM, LDA	テキストデータの構造化
	Blazing text	なし	Word2Vec	センチメント分析
		あり	Text Classification	単語のマイニング
	Object2Vec	あり	Word2Vec 一般ベクトル化	分類，レコメンド
時系列	DeepAR Forecasting	あり	Autoregressive RNN	確率的な時系列予測
異常検知	IP Insights	なし	NN (IPとentityの関連付け)	悪意あるIPアドレスの検出

https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

Image Classification

- ILSVRC 2015で優勝したResNet による高精度な画像認識
- AWS が学習済みのモデルを提供しており、ユーザ固有のデータに合わせて追加学習 (転移学習) が可能

dog

cat

https://docs.aws.amazon.com/ja_jp/sagemaker/latest/dg/image-classification.html

ビルトインアルゴリズムを利用した場合のコード

- 使いたいアルゴリズムのコンテナイメージを呼び出す
- S3のデータを呼び出して fit すれば学習が行われる
- インスタンス数を指定するだけで，分散学習がすぐに実行できる

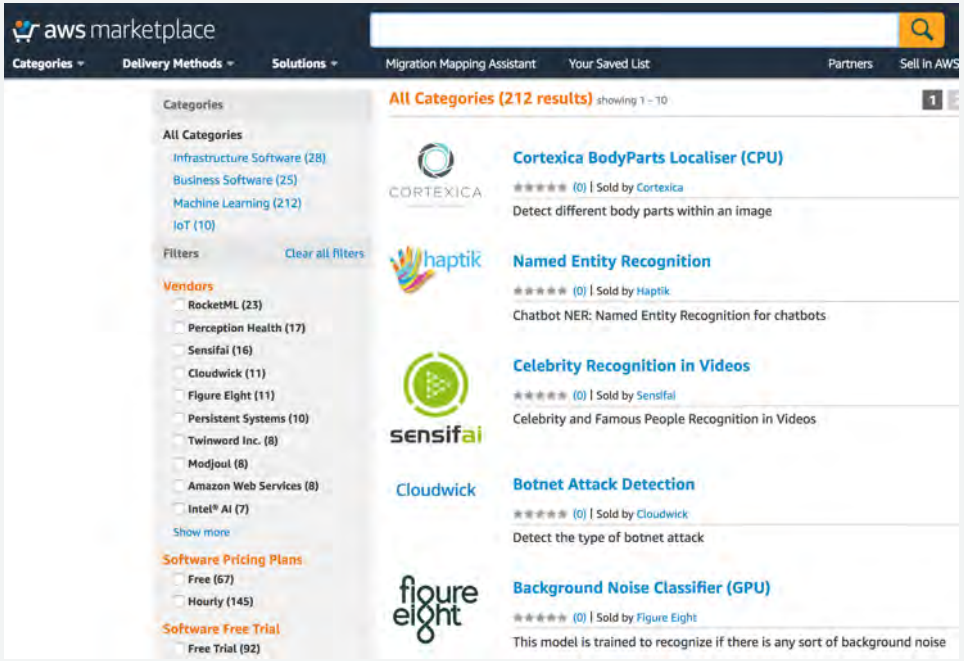
```
from sagemaker.estimator import Estimator

estimator = Estimator(container,
                      train_instance_count=1,
                      train_instance_type='ml.c4.xlarge')

estimator.fit({'train': s3_train_data})
```

アルゴリズムに対応したコンテナ

学習の実行

AWS Marketplace から機械学習のモデルを購入する

- AWSマーケットプレイス経由で，SageMaker上で使う機械学習モデルの売買が可能．小売、メディア向けなど200 以上のアルゴリズムがすでに公開済み



アルゴリズム購入者：

Amazon SageMaker で学習ジョブおよび，推論エンドポイント（バッチ推論ジョブもok）

アルゴリズム購買者：

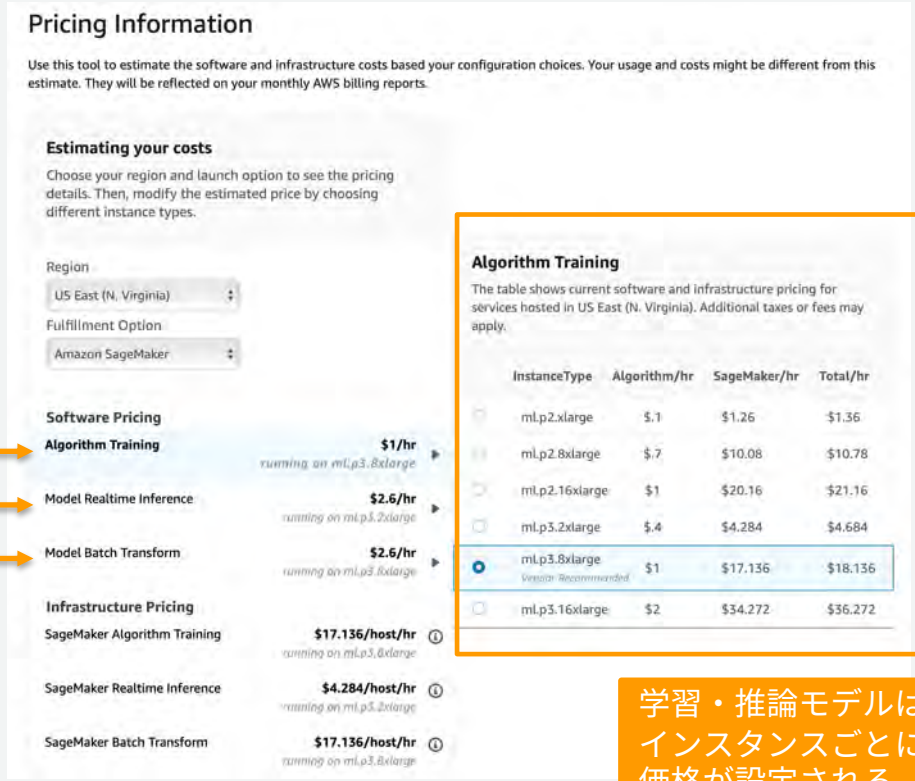
モデルの中身を秘匿してモデルの出品が可能

MLマーケットプレイス 価格

SageMakerコンソール，SDK，AWS CLIからアクセス可能

価格：モデルパッケージの利用に対し，ソフトウエア使用料と AWS のリソース使用料を利用時間に応じて課金

- アルゴリズム学習
- モデルリアルタイム推論
- モデルバッチ推論



学習・推論モデルはインスタンスごとに価格が設定される

開発・学習・推論を効率化する基本機能

Amazon S3　　Amazon SageMaker　　Amazon ECR

学習データ

学習スクリプト

DL / ML
実行環境

Tensor　PYTORCH　mxnet　Chainer　learn

よく使われる機械学習
環境をコンテナで提供

様々なコンテナイメージの提供

- 一般的に利用される機械学習のフレームワークは、コンテナイメージとして提供されており、ユーザは必要なものを選んで利用できる

- フレームワークごとの Docker ファイルは全てgithubで公開
- ユーザーは公開された Docker を自由にカスタムし，自身のコンテナをECR に上げることでカスタム環境を SageMaker で利用できる

aws

Amazon SageMaker でサポートしているフレームワーク一覧

	フレームワーク	SageMaker container サポートバージョン
Deep learning	TensorFlow	**Legacy mode:** 1.4.1, 1.5.0, 1.6.0, 1.7.0, 1.8.0, 1.9.0, 1.10.0 **Script mode:** 1.11.0, 1.12.0, 1.13.0, 1.14.0 **RL:** Coach 0.10.1, 0.11.1 with TF 0.11.0
	Chainer	4.0.0, 4.1.0, 5.0.0
	PyTorch	0.4.0, 1.0.0, 1.1.0, 1.2.0
	MXNet	0.12.1, 1.0.0, 1.1.0, 1.2.1, 1.3.0, 1.4.0, 1.4.1 **RL:** Coach 0.10.1, 0.11.1
ML	scikit-learn	0.20.0
SparkML serving	Spark	1.1 (MLeap version – 0.9.6)

https://github.com/aws/sagemaker-python-sdk#mxnet-sagemaker-estimators
TensorFlow https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/tensorflow
Chainer: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/chainer
PyTorch: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/pytorch
MXNet: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/mxnet
Sklearn: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/sklearn

※ 2019年10月29日時点

SageMaker コンテナによる学習と推論のメリット

学習

- 分散学習を簡単に行うことができる
- ハイパーパラメータ最適化機能を提供，並列実行可能
- コンテナが動く環境であれば手元のPCでも実行可 (ローカルモード)

推論

- リアルタイム推論とバッチ推論を利用可能
- 適切なGPUをアタッチできるElastic Inferenceによる推論高速化
- オートスケーリング, A/Bテスト可能なエンドポイントを簡単に構築

aws

分散学習が簡単に実装できる

インスタンス間の通信などインフラ部分は実装済

- 学習用APIへのリクエスト

インスタンスタイプと
インスタンス数を指定

```
estimator = TensorFlow(entry_point='train.py',
                       train_instance_type='ml.p3.2xlarge',
                       train_instance_count=2,
                       )
```

- 学習コード内の実装 (例: TensorFlowでHorovodを利用)

コード自体は分散学習
対応で書く必要がある

```
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())
opt = hvd.DistributedOptimizer(opt)
```


aws

ハイパーパラメータ最適化

- Deep Learning では、多くのパラメータを調整する必要がある
  - 学習率
  - Dropout率
  - バッチサイズ, ...
- 学習ジョブの並列実行を活用し、最適なパラメータを高速に探索
- デフォルトはベイズ最適化
- ランダム探索も選択可能に NEW!
- HPO ウォームスタートも利用可能

ハイパーパラメータ最適化 (HPO) の書き方

1. Chainer estimator 初期化の際に hyperparameter を指定
2. hyperparameter_ranges に探索したいハイパーパラメータを連続値またはリストで範囲指定
3. Tunerジョブを指定，Hyperparameter_rangesや最適化の目的となるメトリクスを定義する
4. .fit() でベイズ最適化によるHPOジョブスタート．

```
from sagemaker.chainer import Chainer

estimator = Chainer(entry_point="mnist.py",
                    role=role,
                    framework_version='5.0.0',
                    train_instance_count=1,
                    train_instance_type='ml.m4.xlarge',
①                   hyperparameters={'epochs': 30, 'batch-size': 256})

hyperparameter_ranges = {'lr': ContinuousParameter(0.001, 0.1),
②                        'batch-size': CategoricalParameter([32,64,128,256,512])}

tuner = HyperparameterTuner(estimator,
                            objective_metric_name,
③                           hyperparameter_ranges,
                            metric_definitions,
                            max_jobs=9,
                            max_parallel_jobs=3,
                            objective_type=objective_type)

tuner.fit({'training': inputs})
```

https://github.com/aws/sagemaker-python-sdk#sagemaker-automatic-model-tuning
https://aws.amazon.com/jp/blogs/news/amazon-sagemaker-automatic-model-tuning-becomes-more-efficient-with-warm-start-of-hyperparameter-tuning-jobs/

ローカル環境での実行

1. Notebook Instance の上で学習・推論を行う
   - デバッグの際，Dockerから学習・推論インスタンスを立ち上げる時間が勿体無い
   - train_instance_type='local' とする，Notebook Instance のインスタンスサイズをデバッグ用途に合わせる
2. オンプレミス環境でSageMakerを動かす
   - Docker，AWS SDK，SageMaker Python SDK をインストール SageMakerの学習・推論ジョブを実行可能
   - オンプレミスとのハイブリッドなSageMaker環境が作成できる
   - train_instance_type='local_gpu' でローカルGPUで学習を実行

https://aws.amazon.com/jp/blogs/news/sagemaker_from_onpremises/

SageMaker コンテナによる学習と推論のメリット

学習

- 分散学習を簡単に行うことができる
- 並列実行可能でハイパーパラメータ最適化も可能
- コンテナが動く環境であれば手元のPCでも実行可 (ローカルモード)

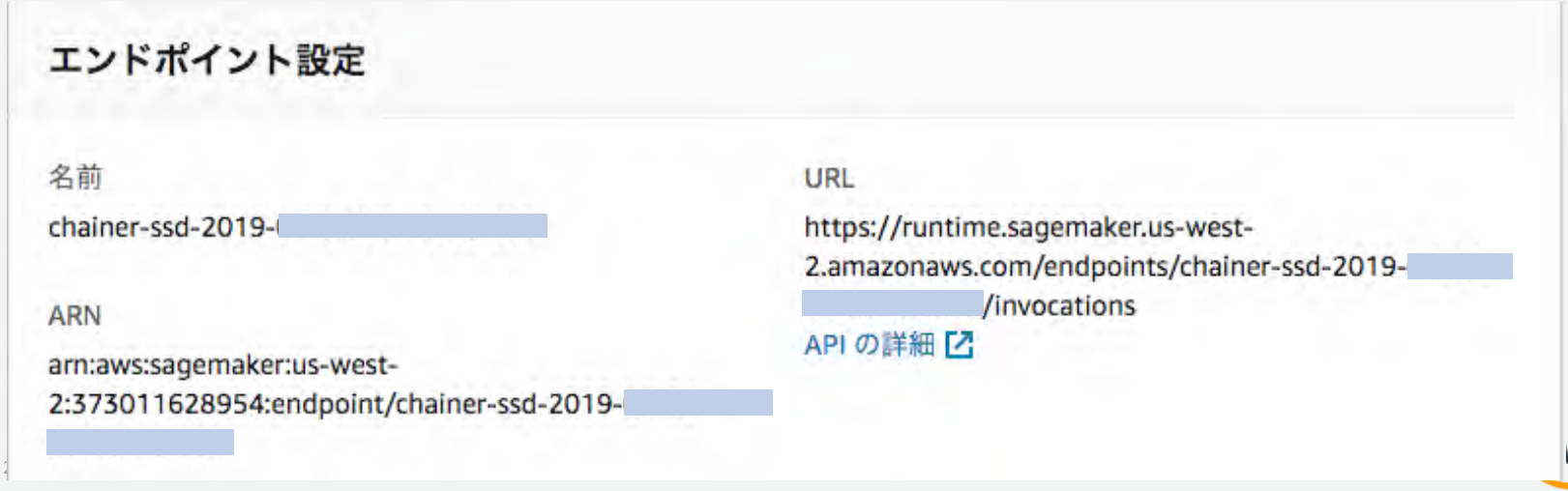
推論

- リアルタイム推論とバッチ推論を利用可能
- 適切なGPUをアタッチできるElastic Inferenceによる推論高速化
- オートスケーリング, A/Bテスト可能なエンドポイントを簡単に構築

aws

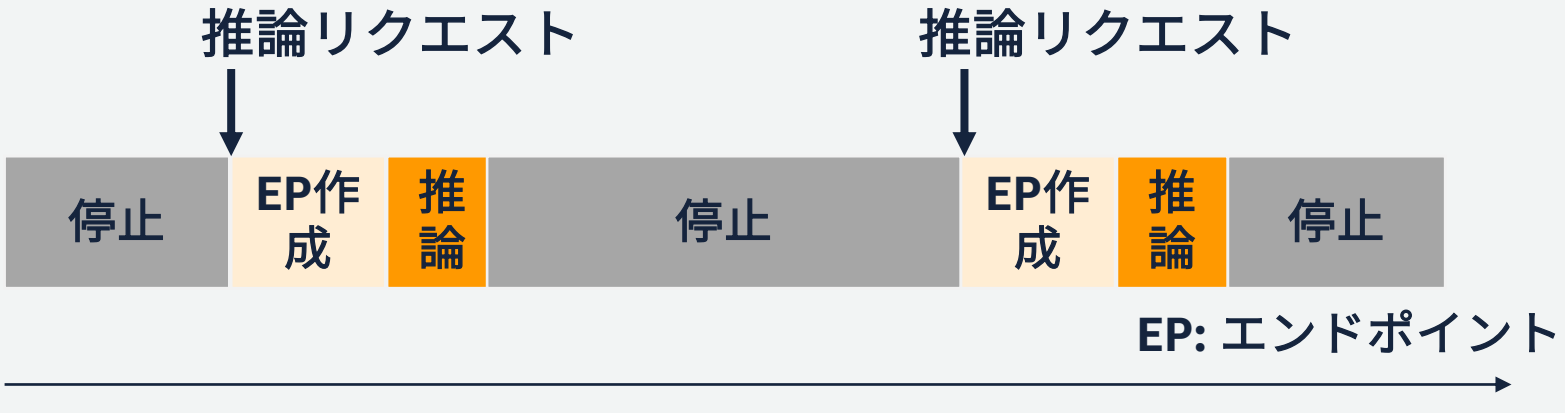
リアルタイム推論

- deploy() を呼ぶだけでエンドポイントを構築でき、APIサーバ構築に関する作業は不要
- Web APIのURLが発行され、URL にリクエストを送ると推論できる
- エンドポイント起動中は料金がかかる



バッチ変換ジョブ (バッチ推論)

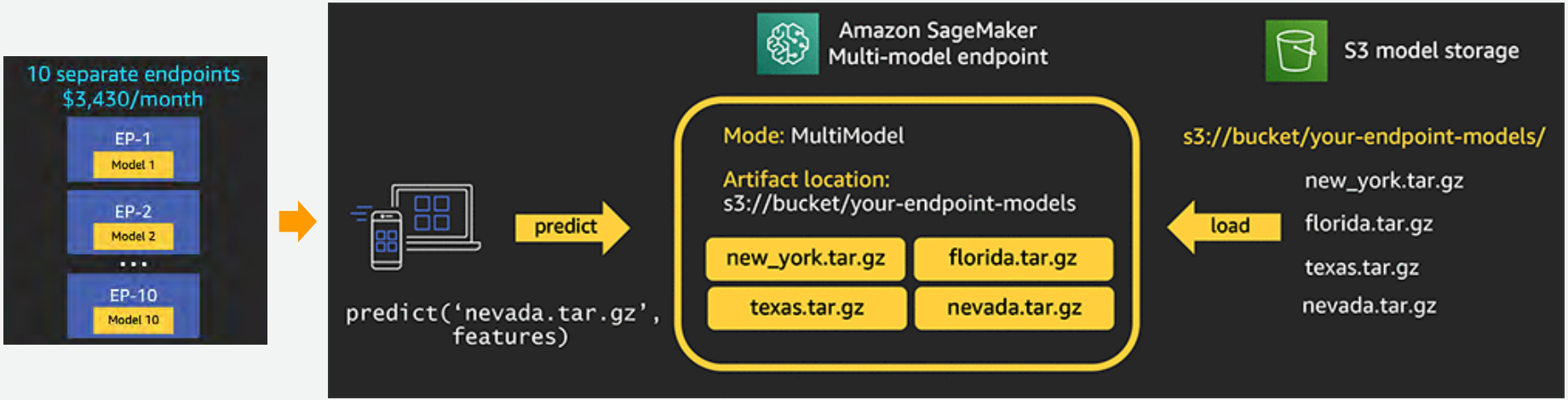
- リアルタイム推論が必要でない場合、推論エンドポイントを維持するとコストがかかる
- バッチ推論では、推論が必要なときに、エンドポイントを作成し、推論後のエンドポイント削除を自動で行う



マルチモデルエンドポイント(MME)が利用可能に

NEW!

- 一つのエンドポイントの裏で複数のモデルを切り替えて推論コストを削減
- 学習済みモデルは S3にストアしてエンドポイントを動的に切り替え可能に



https://aws.amazon.com/blogs/machine-learning/save-on-inference-costs-by-using-amazon-sagemaker-multi-model-endpoints/

推論に最適な G4 / R5 インスタンスをサポート

- デプロイ時に、リアルタイム推論用の G4 および R5 インスタンスを選択可能に
- G4 (NVIDIA T4 Tensor Core GPU)
    - GPU の低レベルのソフトウェアライブラリにアクセスする必要がある、画像分類、オブジェクト検出、推奨エンジン、自動音声認識、言語翻訳といった機械学習アプリケーションのデプロイのために最適化
- R5 (Intel® Xeon® スケーラブル (Cascade Lake) プロセッサ)
    - メモリとコンピューティングの要件に基づいた、適切なサイズのインスタンスにより、お客様がコストを削減できるようにする、EBS 最適化バーストのサポートを備えたメモリ最適化インスタンス

https://aws.amazon.com/jp/about-aws/whats-new/2019/10/amazon-sagemaker-supports-g4-r5-instances-real-time-inference/

aws

CPU インスタンスの計算を GPU でアクセラレート

- スタンドアロンのGPU は主に学習に最適化されており、推論には大きすぎる
- 適切なGPU リソースをCPUに関連づけることで，高速な推論を低コストで実行

インスタンス	EIA	推論速度 (msec)	価格 ($/hour)
c5.large	なし	230 msec	$0.085
c5.large	eia1.medium	46 msec	$0.22
p2.xlarge	なし	42 msec	$0.90

価格は 2018/11 時点のバージニア北部のもの

https://aws.amazon.com/jp/blogs/news/amazon-elastic-inference-gpu-powered-deep-learning-inference-acceleration/

Amazon Elastic Inference

- 推論に適した低コストのGPU駆動のアクセラレーションを，CPU の EC2 および SageMaker インスタンスに適用．DL 実行コストを最大75%削減

- **利用方法**
  - エンドポイントのインスタンスにEIAアタッチして利用
  - ローカルモード推論時は SageMaker Notebook インスタンスにEIAをアタッチ

Accelerator type	TFLOPS FP32 throughput	TFLOPS FP16 throughput	Memory in GB
ml.eia1.medium	1	8	1
ml.eia1.large	2	16	2
ml.eia1.xlarge	4	32	4

https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

機械学習パイプライン全体の支援

- 2018年のre:Inventで、開発・学習・推論に対する基本となる機能に加えて、ラベリングやモデル変換をサポート
- パイプライン全体を管理する機能も追加

機械学習パイプライン全体の支援

- 2018年のre:Inventで、開発・学習・推論に対する基本となる機能に加えて、ラベリングやモデル変換をサポート
- パイプライン全体を管理する機能も追加

Git 統合 によりレポジトリからの直接学習が可能に

- SageMaker 上で Git レポジトリを登録しておくことで、ノートブックインスタンス起動時に clone される
- Amazon SageMaker Python SDK で学習/ホスティングする際に Git リポジトリからスクリプトを直接指定可能に

```
git_config = {'repo': 'https://github.com/awslabs/amazon-sagemaker-examples.git',
                                                    'branch': 'training-scripts'}
estimator = TensorFlow(entry_point='train.py',
                       source_dir='char-rnn-tensorflow',
                       git_config=git_config,
                       train_instance_type=train_instance_type,
                       train_instance_count=1,
                       role=sagemaker.get_execution_role(), # このノートブックで使用し
                       framework_version='1.13',
                       py_version='py3',
                       script_mode=True)
```

NEW!

https://aws.amazon.com/jp/blogs/news/git-integration-now-available-for-amazon-sagemaker-python-sdk/

aws

機械学習パイプライン全体の支援

- 2018年のre:Inventで、開発・学習・推論に対する基本となる機能に加えて、ラベリングやモデル変換をサポート
- パイプライン全体を管理する機能も追加

学習ジョブの検索機能

NEW!

- 学習ジョブ名や学習時に指定したタグなどで、過去に実施した学習ジョブを検索できる

- 学習時の精度を正規表現で記録していれば、コンソール上でソートが可能で、最も良い学習結果を後で探すことができる

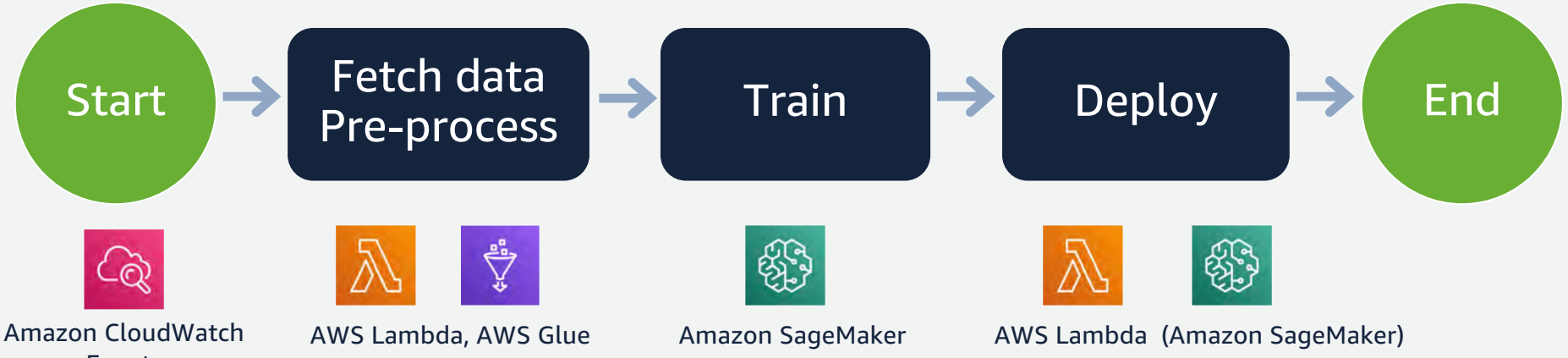
https://aws.amazon.com/jp/blogs/machine-learning/amazon-sagemaker-now-comes-with-new-capabilities-for-accelerating-machine-learning-experimentation/?nc1=h_ls

aws

AWS Step Functions Data Science SDK

NEW!

- AWS Lambda, Glue などにも対応したサーバーレスオーケストレーション
- AWS Step Functions Data Science SDK が登場、Pythonで前処理～学習～デプロイ～のデータサイエンスワークフローを作成し、視覚化できます



https://aws.amazon.com/jp/about-aws/whats-new/2019/11/introducing-aws-step-functions-data-science-sdk-amazon-sagemaker/

aws

Amazon Athena で SQL クエリから ML 実行

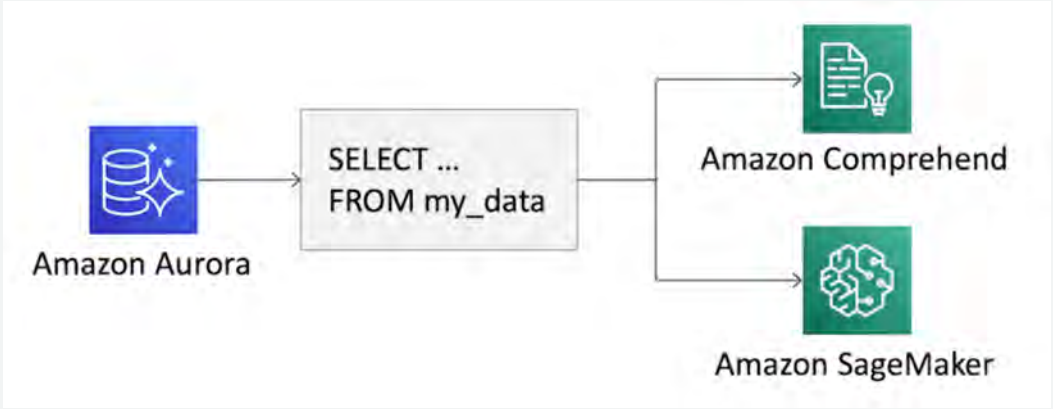
- Athena を使って SQL クエリから直接推論を呼び出し可能に。
- 機械学習モデルをコンソール、API、JDBCドライバからに呼び出し可能
- SageMaker で学習したモデルやMarketplaceで購入したモデルをSageMaker hosting service としてデプロイ、AthenaからSQLで呼び出し

https://aws.amazon.com/about-aws/whats-new/2019/11/amazon-athena-adds-support-for-invoking-machine-learning-models-in-sql-queries/

Amazon Aurora Machine Learning

NEW!

- SQL言語を使用して、データベースに機械学習の予測機能を追加
- Auroraデータベースから、Amazon SageMaker および Amazon Comprehend へ統合
- Amazon SageMaker モデルトレーニングのために、Amazon Aurora からデータを S3 にエクスポート




```sql
CREATE FUNCTION will_churn (
        state varchar(2048), acc_length bigint(20),
        area_code bigint(20), int_plan varchar(2048),
        vmail_plan varchar(2048), vmail_msg bigint(20),
        day_mins double, day_calls bigint(20),
        eve_mins double, eve_calls bigint(20),
        night_mins double, night_calls bigint(20),
        int_mins double, int_calls bigint(20),
        cust_service_calls bigint(20))
RETURNS varchar(2048) CHARSET latin1
        alias aws_sagemaker_invoke_endpoint
        endpoint name 'estimate_customer_churn_endpoint_version_123';
```

例）SQLでSageMakerエンドポイントにアクセスして、顧客離反予測

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-ml.html

aws

Apache Airflow 対応

- Airflow 1.10.1 よりSageMaker Operator をサポート
- 既存の Airflow 環境からSageMaker を呼んでパイプラインを構築可能

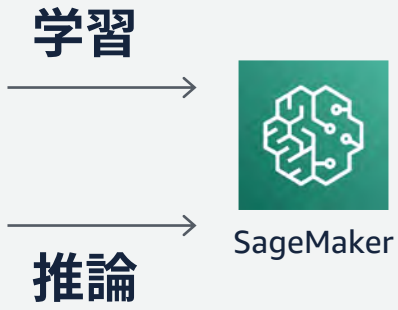
Airflowサーバでの設定

Airflow の Operator を利用して
DAG でパイプラインを定義

```
train_op =
        SageMakerTrainingOperator(…)
transform_op =
        SageMakerTransformOperator(…)

transform_op.set_upstream(train_op)
```

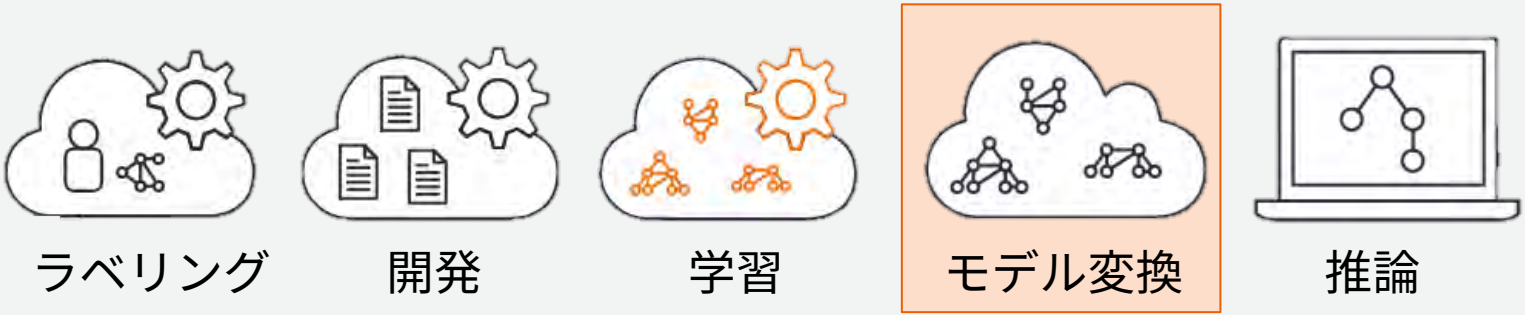
SageMakerのAPIで
学習・推論の設定作成

```
train_config = training_config(…)

trans_config
=transform_config_from_estimator(…
)
```



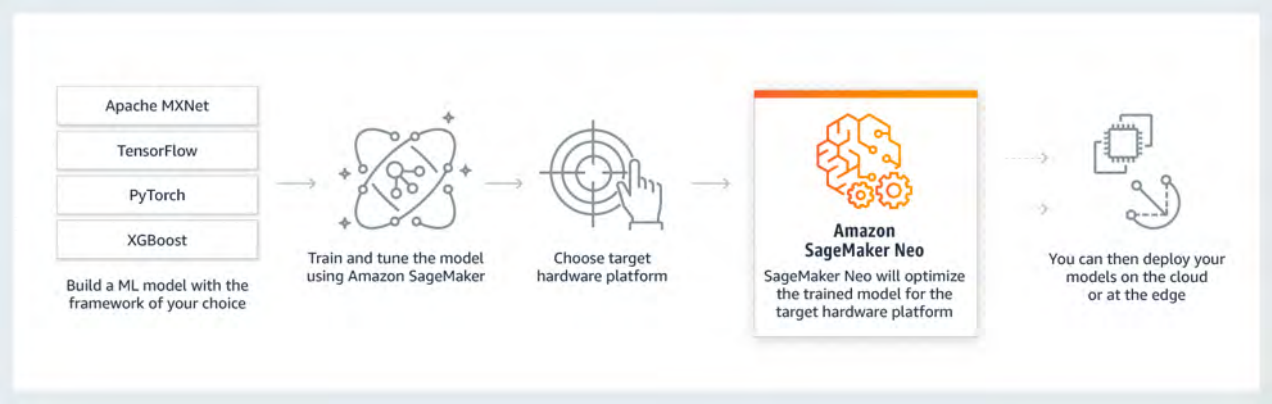
機械学習パイプライン全体の支援

- 2018年のre:Inventで、開発・学習・推論に対する基本となる機能に加えて、ラベリングやモデル変換をサポート
- パイプライン全体を管理する機能も追加



Amazon SageMaker Neo

- Tensorflow や PyTorch などのモデルを、 EC2 インスタンスや Greengrass デバイス上で高速に動作するように変換するサービス
- 従来のDeep Learning フレームワークが 500MB-1GB 程度であるのに対し、Amazon SageMaker Neo Runtime は 1MB 程度
- Apache Software License で OSS として提供予定



SageMaker Python SDK による Neo 利用の流れ

学習

C5向けに最適化

C5にデプロイ

```
mnist_estimator = TensorFlow(
                entry_point='mnist.py', role=role, framework_version='1.11.0',
                    training_steps=1000, evaluation_steps=100,
                train_instance_count=2, train_instance_type='ml.c4.xlarge')

mnist_estimator.fit(inputs)

optimized_estimator = mnist_estimator.compile_model(
                target_instance_family='ml_c5', input_shape={'data':[1, 784]},
                    output_path=output_path, framework='tensorflow',
                    framework_version='1.11.0')

optimized_predictor = optimized_estimator.deploy(
                initial_instance_count = 1, instance_type = 'ml.c5.4xlarge')
```


aws

セキュリティ: 暗号化とコンプライアンス

- 学習と推論のジョブにおいてサーバ側の暗号化 (SSE-KMS) を利用可能
- 以下のものをすべて暗号化可能
  - 学習時の入出力データ
  - 学習用インスタンス、エンドポイントインスタンスのストレージ
  - バッチ推論時の入出力データ
- Cloudtrail に対応済み
- PCI DSS および HIPPAに対応済み

セキュリティ: 閉域網での通信

- SageMaker と S3 のデータ通信はすべて S3 VPC エンドポイント経由で行うことが可能
  - 学習ジョブの入出力における S3 アクセス
  - 学習済モデルをデプロイする際の S3 アクセス
- SageMaker の API はすべて PrivateLink 経由で行うことが可能
  - SageMaker Notebook Endpoint
  - SageMaker Service API
  - SageMaker Runtime API

料金について データサイエンティストが 1 週間利用した際の料金の利用例

時間数	ノートブック インスタンス	トレーニング インスタンス	ホスティング インスタンス	費用: 1 時間あたり USD	合計
105.00	ml.t2.medium			0.0464 USD	4.872 USD
2.00		ml.m4.4xlarge トレーニング		1.12 USD	2.24 USD
0.67			ml.t2.medium	0.065 USD	0.04355 USD
					7.1555 USD

入力処理 GB – ノートブック	出力処理 GB – ノートブック	入力データ GB – ホスティング	出力データ GB – ホスティング	入力または出力 GB あたり料金	合計
4 * 0.1 = 0.4 GB	4 * 1 = 4 GB	4 * 1 = 4 GB	4 * 0.1 = 0.4 GB	0.016 USD	0.1408 USD

アジェンダ

- 機械学習の課題に対するSageMakerのメリット
- SageMakerを利用した機械学習プロセス
- 機械学習の開発・学習・推論を効率化する基本機能
- まとめ

まとめ

- 機械学習のマネージドサービス Amazon SageMaker を使うことで，インフラ構築を気にせず，すぐに機械学習をはじめることができる
- DLフレームワークやライブラリの定常的なアップデートを気にすることなく，モデル開発に集中できる
- 全体のワークフローの管理や，コスト最適のための新サービスなど，便利な機能が日々アップデート

TensorFlow

Using TensorFlow with SageMaker Python SDK

https://sagemaker.readthedocs.io/en/stable/using_tf.html

TensorFlow SageMaker Estimators and Models

https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/tensorflow

SageMaker TensorFlow containers repositories

Training: https://github.com/aws/sagemaker-tensorflow-container

Serving: https://github.com/aws/sagemaker-tensorflow-serving-container

aws

MXNet

Using MXNet with SageMaker Python SDK

https://sagemaker.readthedocs.io/en/stable/using_mxnet.html

MXNet SageMaker Estimators and Models

https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/mxnet

SageMaker MXNet containers repositories

Training: https://github.com/aws/sagemaker-mxnet-container

Serving: https://github.com/aws/sagemaker-mxnet-serving-container

aws

PyTorch

Using PyTorch with SageMaker Python SDK
https://sagemaker.readthedocs.io/en/stable/using_pytorch.html

PyTorch SageMaker Estimators and Models
https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/pytorch

SageMaker PyTorch containers repository
https://github.com/aws/sagemaker-pytorch-container

aws

Chainer

Using Chainer with SageMaker Python SDK
https://sagemaker.readthedocs.io/en/stable/using_chainer.html

Chainer SageMaker Estimators and Models
https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/chainer

SageMaker Chainer containers repository
https://github.com/aws/sagemaker-chainer-container

Reference

SageMaker Example Notebooks

https://github.com/awslabs/amazon-sagemaker-examples

SageMaker SDK

https://github.com/aws/sagemaker-python-sdk
(Doc はこちら: https://readthedocs.org/projects/sagemaker/)

SageMaker 公式ドキュメント

https://docs.aws.amazon.com/ja_jp/sagemaker/latest/dg/whatis.html