7. 세이지메이커 노트북 인스턴스

2강. 학습 모델 구축하기

학습목표

- 세이지메이커 노트북 인스턴스를 이용하여 머신러닝 학습 모델을 구축하는 단계를 이해할 수 있다.

학습내용

- Amazon S3 버킷 생성
- 데이터 생성
- 모델 훈련
- 모델 배포 및 검증
- 정리

1. Amazon S3 버킷 생성

- S3 버킷 폴더 생성
  - ✓ S3 콘솔 접속 후 버킷 이름 클릭
  - ✓ [폴더 만들기] 을 클릭하여 폴더 생성

- 폴더 이름 입력 후 저장 클릭

- 버킷/폴더 생성 완료
- 폴더 경로 : S3/Bucket name / folder

2. 데이터 생성

- 세이지메이커 노트북 인스턴스 생성 주피터 노트북 생성 단계

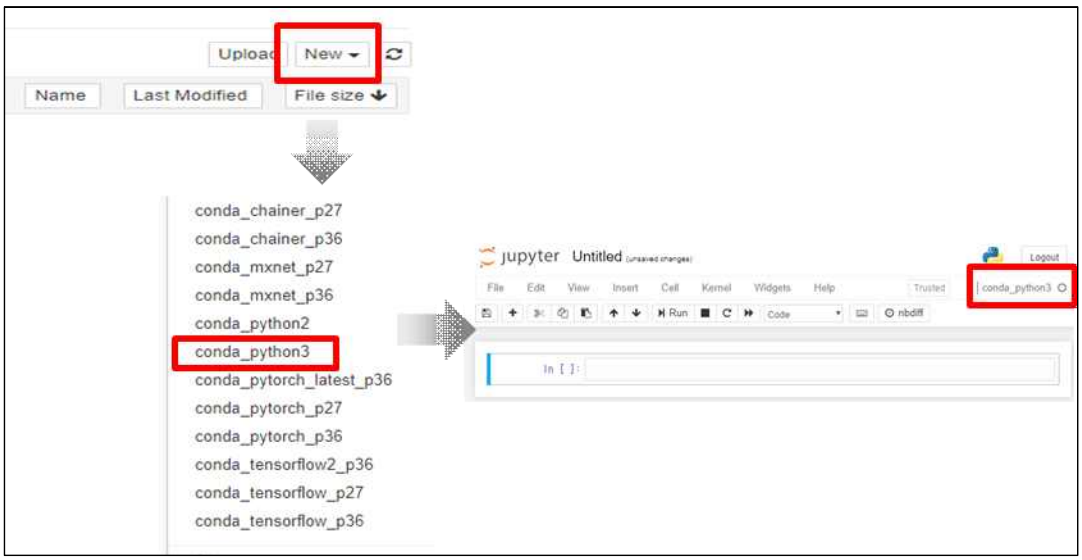
① MNIST 데이터셋 다운로드

② 훈련 데이터셋 탐색

③ 훈련 데이터셋 변환 및 S3에 업로드

✓ 주피터 노트북 생성

- 파일 탭에서 새로 만들기를 선택하고 conda_python3를 선택
- 사전 설치된 환경에는 기본 Anaconda 설치,Python3이 포함되어 있음



✓ MNIST 데이터셋 다운로드

- MNIST Database 웹 사이트에서 노트북으로 데이터셋을 다운로드
- 데이터셋 파일명 : mnist.pkl.gz

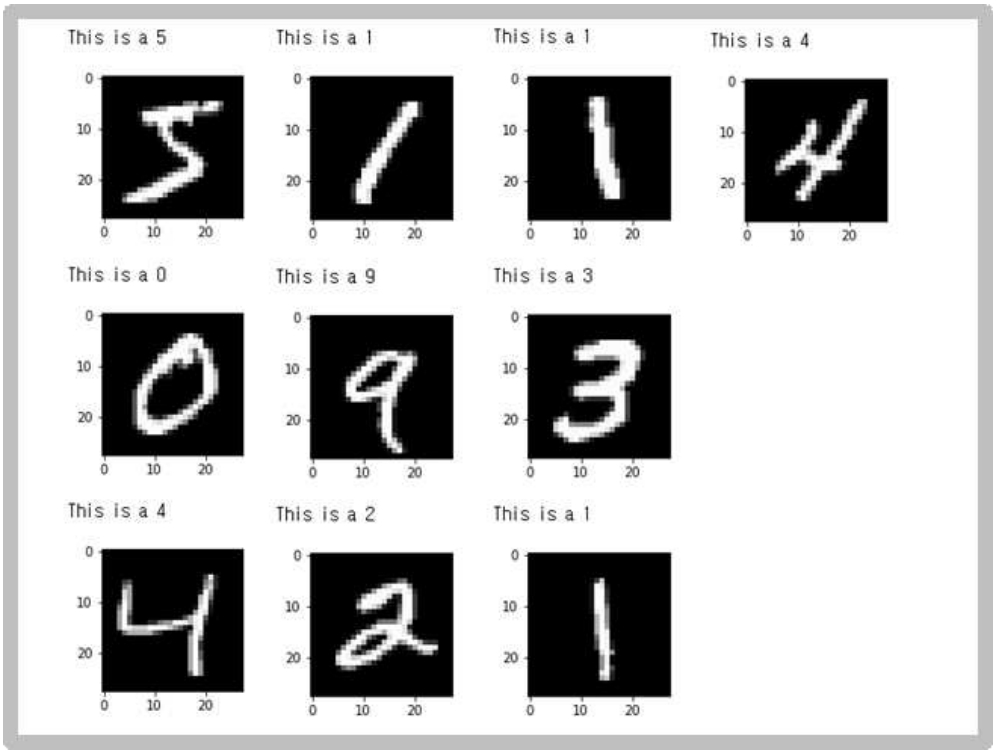
```
%%time
import pickle, gzip, urllib.request, json
import numpy as np
# Load the dataset
urllib.request.urlretrieve("http://deeplearning.net/data/mnist/mnist.pkl.gz", "mnist.pkl.gz")
with gzip.open('mnist.pkl.gz', 'rb') as f:
    train_set, valid_set, test_set = pickle.load(f, encoding='latin1') print(train_set[0].shape)
```

✓ 훈련 데이터셋 탐색

- Train_set에서 처음 10개의 이미지를 표시

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (2,10)
for i in range(0, 10):
    img = train_set[0][i]
    label = train_set[1][i]
    img_reshape = img.reshape((28,28))
    imgplot = plt.imshow(img_reshape, cmap='gray')
    print('This is a {}'.format(label))
    plt.show()
```

- Train_set에서 처음 10개의 이미지 출력 결과



✓ 훈련 데이터셋 변환 및 S3에 업로드

- 라이브러리 선언

```
%%time
import os
import boto3
import re
import copy
import time
import io
import struct
from time import gmtime, strftime
from sagemaker import get_execution_role
```

- 규칙, 리전, 버킷이름, 데이터를 저장하는 버킷 경로
- myBucket = 생성된 버킷 이름, myFolder= 버킷에 생성한 폴더 이름

```
role = get_execution_role()

region =
boto3.Session().region_name

bucket='myBucket'
prefix = myFolder'
```

- 데이터셋 형식을 numpy.array 형식에서 CSV 형식으로 변환

```
def convert_data():
    data_partitions = [('train', train_set), ('validation', valid_set), ('test',
test_set)]
    for data_partition_name, data_partition in data_partitions:
        print('{}: {} {}'.format(data_partition_name,
data_partition[0].shape,
        data_partition[1].shape))
        labels = [t.tolist() for t in data_partition[1]]
        features = [t.tolist() for t in data_partition[0]]
        if data_partition_name != 'test':
            examples = np.insert(features, 0, labels, axis=1)
        else: examples = features
```

```
np.savetxt('data.csv', examples, delimiter=',')

        key = "{}/{}/examples".format(prefix,data_partition_name)
        url = 's3://{}/{}'.format(bucket, key)

boto3.Session().resource('s3').Bucket(bucket).Object(key).upload_file('
data.csv')
        print('Done writing to {}'.format(url))
convert_data()
```

```
<class 'numpy.ndarray'>
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[5 0 4 ... 8 4 8]
50000 50000
```

numpy.array

	0	1	2	3	4	5	6	7	8	9	...
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

CSV

3. 모델 훈련

✓ 훈련 작업 생성 및 실행

- Amazon SageMaker Python SDK 및 XGboost 컨테이너를 가져옴

```
import sagemaker

from sagemaker.amazon.amazon_estimator import
get_image_uri

container =
get_image_uri(boto3.Session().region_name, 'xgboost')
```

- 데이터를 업로드한 s3 위치에서 훈련 및 검증 데이터를 다운로드하고 훈련 출력을 저장할 위치를 설정

```
train_data = 's3://{}/{}/{}'.format(bucket, prefix, 'train')

validation_data = 's3://{}/{}/{}'.format(bucket, prefix,
'validation')

s3_output_location = 's3://{}/{}/{}'.format(bucket,
prefix, 'xgboost_model_sdk')print(train_data)
```

- sagemaker.estimator.Estimator 클래스의인스턴스를 생성

```
xgb_model = sagemaker.estimator.Estimator(container,
                                          role,
                                          train_instance_count=1,

train_instance_type='ml.m4.xlarge',
                                          train_volume_size = 5,
                                          output_path=s3_output_location,

sagemaker_session=sagemaker.Session())
```

role	Amazon SageMaker가 수임하여 사용자를 대신해 작업을 수행하는 AWS Identity and Access Management(IAM) 역할
train_instance_count , train_instance_type	모델 훈련에 사용할 ML 컴퓨팅 인스턴스의 유형 및 수
train_volume_size	훈련 인스턴스에 연결할 Amazon Elastic Block Store(Amazon EBS) 스토리지 볼륨의 크기(GB)
output_path	Amazon SageMaker이 훈련 결과를 저장하는 S3 버킷의 경로
sagemaker_session	Amazon SageMaker API 및 훈련 작업이 사용하는 다른 모든 AWS 서비스와의 상호 작용을 관리하는 세션 객체

- set_hyperparameters 메서드를 호출하여 XGBoost 훈련 작업의 하이퍼 파라미터 값을 설정

```
xgb_model.set_hyperparameters(max_depth = 5,
                              eta = .2,
                              gamma = 4,
                              min_child_weight = 6,
                              silent = 0,
                              objective = "multi:softmax",
                              num_class = 10,
                              num_round = 10)
```

max_depth	트리의 최대 깊이
eta	과적합 방지 업데이트에 사용되는 단계 크기 축소
gamma	트리의 리프 노드에 추가 파티션을 생성하는 데 필요한 최소 손실 감소
min_child weight	하위에 필요한 인스턴스 가중치(헤시안)의 최소 합
silent	0은 실행 메시지 출력, 1은 자동 모드를 의미
objective	학습 작업 및 해당 학습 목표를 지정
num_class	클래스의 수
num_round	교육을 실행할 라운드의 수

- 훈련 작업에 사용할 훈련 채널을 생성

```
train_channel = sagemaker.session.s3_input(train_data,
content_type='text/csv')
valid_channel = sagemaker.session.s3_input(validation_data,
content_type='text/csv')
data_channels = {'train': train_channel, 'validation': valid_channel}
```

- 모델 훈련을 시작하려면 예측기의 fit 메소드를 호출

```
xgb_model.fit(inputs=data_channels,  logs=True)
```

4. 모델 배포 및 검증

✓ 모델 배포

- deploy 메서드가 배포 가능한 모델을 생성하고, Amazon SageMaker 호스팅 서비스 엔드포인트를 구성하고, 모델을 호스팅할 엔드포인트를 시작
- deploy 메서드를 호출할 때 엔드포인트를 호스팅할 때 사용할 ML 인스턴스의 유형 및 수를 지정

```
xgb_predictor = xgb_model.deploy(initial_instance_count=1,
                                 content_type='text/csv',
                                 instance_type='ml.t2.medium'
                                 )
```

- 엔드포인트 구성

✓ 모델 검증

- Amazon S3에서 테스트 데이터를 다운로드

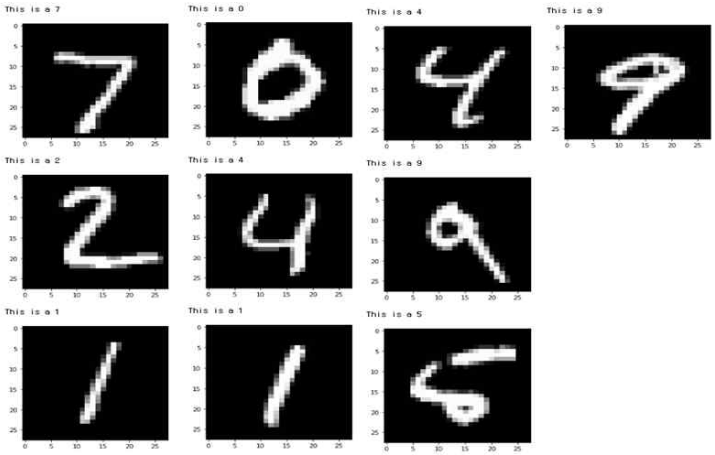
```
s3 = boto3.resource('s3')

test_key = "{}/test/examples".format(prefix)

s3.Bucket(bucket).download_file(test_key, 'test_data')
```

- 테스트 데이터셋의 처음 10개 이미지를 레이블로 플롯

```
%matplotlib inline
for i in range (0, 10):
    img = test_set[0][i]
    label = test_set[1][i]
    img_reshape = img.reshape((28,28))
    imgplot = plt.imshow(img_reshape, cmap='gray')
    print('This is a {}'.format(label))
    plt.show()
```



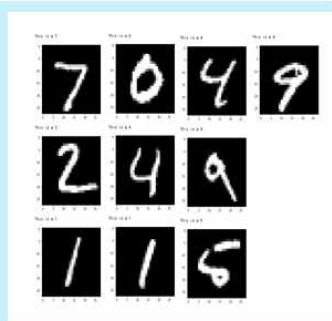
- 테스트 데이터셋의 처음 10개 예제에 대해 추론을 가져오기
- sagemaker.predictor.RealTimePredictor 객체의 predict 메서드를 호출

```
with open('test_data', 'r') as f:
    for j in range(0,10):
        single_test = f.readline()
        result = xgb_predictor.predict(single_test)
        print(result)
```

-모델이 정확하게 예측하는지 확인하려면 이전 단계에서 플롯한 숫자와 비교

```
with open('test_data', 'r') as f:
    for j in range(0,10):
        single_test = f.readline()
        result = xgb_predictor.predict(single_test)
        print(result)
```

```
b'7.0'
b'2.0'
b'1.0'
b'0.0'
b'4.0'
b'1.0'
b'4.0'
b'9.0'
b'5.0'
b'9.0'
```



5. 정리

✓ 정리 단계

- 불필요한 비용이 발생하지 않도록 AWS Management 콘솔을 사용하여 연습에서 생성한 리소스를 삭제

엔드 포인트	엔트포인트를 삭제하려면 ML 컴퓨팅 인스턴스 또는 이를 지원하는 인스턴스도 삭제해야 함 ① 추론에서 엔드포인트를 선택 ② 예제에서 생성한 엔드포인트를 선택한 다음 action(작업)Delete(삭제 선택) ③ Delete를 선택

엔드 포인트 구성	① 추론에서 엔드포인트 구성을 선택 ② 예제에서 생성한 엔드포인트 구성을 선택한 다음 action(작업)Delete(삭제 선택) ③ Delete를 선택

모델	① 추론에서 모델을 선택 ② 예제에서 생성한 모델을 선택한 다음action(작업)Delete(삭제 선택) ③ Delete를 선택

노트북 인스턴스 (중지 후 삭제)	① 노트북에서 노트북 인스턴스 선택 ② 예제에서 생성한 노트북 인스턴스를 선택한다음, Action(작업)|Stop(중지)를 선택 ③ 노트북 인스턴스를 중지하는데 몇 분정도 소요 ④ 상태가 Stopped(중지됨)으로 변경 되면Action(작업)/Delete(삭제)를 선택 ⑤ Delete를 선택

S3	① Amazon s3 콘솔을 열고 모델 결과물 및 훈련 데이터셋을 저장하기 위해 생성한버킷을 삭제 ② Amazon CloudWatch 콘솔을 열과이름이 /aws/sagemaker/로 시작하는 모든 로그 그룹을 삭제

평가하기

1. 세이지메이커 노트북 인스턴스를 이용하여 머신러닝 학습 모델을 구축하는 단계를 올바르게 나열하시오.

① 모델 훈련

② Amazon S3 버킷 생성

③ 세이지메이커 노트북 인스턴스 및 주피터 노트북 생성

④ 데이터 생성

⑤ 정리

⑥ 모델 배포 및 검증

- 정답 : ② → ③ → ④ → ① → ⑥ → ⑤

2. 세이메이커 노트북 인스턴스를 사용하여 학습모델을 구축한 후, 불필요한 비용을 발생하지 않도록 정리단계에서 삭제해야 할 리소스의 종류로 관련이 먼 것은?

① 엔드포인트

② EC2

③ 모델

④ 노트북 인스턴스

- 정답 : ②번

해설 : 정리단계에서 엔드포인트, 엔드포인트 구성, 모델, 노트북 인스턴스, S3를 삭제합니다. EC2는 노트북 인스턴스에 포함되어 삭제됩니다.

학습정리

1. Amazon S3 버킷 생성
   - S3 콘솔 접속 후 버킷 이름 클릭
   - [폴더 만들기] 클릭 - 폴더 이름 입력 후 저장

2. 데이터 생성
   - 주피터 노트북 생성 : conda_python3 선택
   - 데이터 생성

     ① MNIST 데이터셋 다운로드 : MNIST Database에서 다운로드

     ② 훈련 데이터셋 탐색 : 데이터 확인

     ③ 훈련 데이터셋 변환 및 S3에 업로드 : CSV로 데이터 형식 변환 및 업로드

3. 모델 훈련
   - 훈련 작업 생성 및 실행 : XGBoost 알고리즘 사용

4. 모델 배포 및 검증
   - 모델 배포 : 배포 모델 생성 및 엔드포인트 구성
   - 모델 검증 : 추론 과정 실행

5. 정리
   - 불필요한 비용이 발생하지 않도록 생성한 리소스 삭제
   - 엔드포인트, 엔드포인트 구성, 모델, 노트북 인스턴스, S3 등 삭제