CPD Lab 6: Machine Learning and Big Data

Introduction

In this lab you will complete a series of tasks relating to AWS machine learning and big data. For machine learning, you will use Rekognition service to obtain image labels. In another task, you will use Textract to extract text from an image that is then read out using Polly service. You will also explore use of SageMaker and Deep Learning AMI for running machine learning models.
For big data, you will use Data Pipeline to transfer data from a DynamoDB table to S3. In another task you will use EMR cluster to run a Hive script to analyse log files.

Tasks:

- Task 1: Using Amazon Rekognition
- Task 2: Amazon Textract and Amazon Polly
- Task 3: Sample chatbot using Amazon Lex
- Task 4: Machine Learning with SageMaker
- Task 5: Machine Learning with Deep Learning AMI
- Task 6: Running a job on Elastic MapReduce (EMR)
- Task 7 : Data Pipeline to export DynamoDB data to S3 bucket
- Task 8: Releasing the allocated resources

Task 1: Using Amazon Rekognition

In this task you will send an image to Rekognition service to find the labels and confidence levels for the image content.

- A (.py) file is available in “CPD Lab 6 resources.zip” to obtain the labels for an image using Amazon Rekognition
- Run the file to obtain the labels by providing an image of your choice

Task 2: Amazon Textract and Amazon Polly

In this task, you will use a Python program to pass an image containing text to Textract service to obtain the textual information from an image. Then in the same program the obtained text will be converted to audio using Polly service and played back.

- The program and an image are in “CPD Lab 6 Resources”
- You might have to wait a bit for the default audio program to start

Note: The automatic playback on your local machine may depend on the default program to play mp3 files. You can check this under Control Panel->Default Programs->set your default programs

Task 3: Sample chatbot using Amazon Lex

Amazon Lex is used to build conversational chatbots using text and voice. In this task you will create a chatbot based on one of the samples for 'OrderFlowers' provided by AWS.

More details are at
https://docs.aws.amazon.com/lex/latest/dg/how-it-works.html

For this task, you will use the 'OrderFlowers' blueprint on AWS

- Navigate to Amazon Lex under services on AWS management console
- Choose 'Get Started' and click 'Create'
- Select 'OrderFlowers' blueprint
- Select 'No' for COPPA
- Click 'Create'
- After the bot has been built, click 'Test
- You can now interact with the bot either by typing text or conversing with the bot

Creating a custom chatbot is also very simple and an exercise is provided at https://docs.aws.amazon.com/lex/latest/dg/getting-started-ex2.html that you can optionally explore.

Task 4: Machine Learning with Sagemaker

In this task you will complete an AWS tutorial provided at
https://aws.amazon.com/getting-started/tutorials/build-train-deploy-machine-learning-model-sagemaker/

In this task, you will follow the instructions in the link above to:

- Enter Amazon SageMaker console
- Create an Amazon SageMaker notebook instance
- Prepare the data
- Train the model to learn from the data
- Deploy the model
- Evaluate your ML model's performance
- Terminate your resources

Task 5: Machine Learning with Deep Learning AMI

The AWS Deep Learning AMI (DLAMI) comes pre-configured with NVIDIA and the popular deep learning frameworks. You can also install your own models.

Step 1: Create a DLAMI

Follow the steps listed below. If further information is required then refer to 'Links' Section at the end of this document:

- Select EC2 from Services in the AWS console
- Click 'Launch Instance'
- **Choose an AMI**
  - Select 'AWS Marketplace' tab on the left
  - Search for 'Deep learning Ubuntu'
  - Select 'Deep Learning AMI (Ubuntu 16.04)'
  - On the 'Product details, page, click 'continue'
- **Choose an Instance Type**
  - A 'p3.2xlarge' instance will be pre-selected but is not available in 'eu-west-2' so select 't2.small' towards the beginning of the list. The instance type does not matter here as we are only interested to get a model to run and then terminate it without waiting for it to finish
  - Click Next
- **Configure Instance Details – click Next**
- **Add Storage – click Next**
- **Add Tags – click Next**
- **Configure Security Group**
  - Let the default selection 'Create a new security group'
  - In the settings for SSH (inbound rules) through port 22, change 'Source' to 'My IP'
  - Click 'Review and Launch'
- Review Instance Launch
  - Click 'Launch'
- In the dialog box that pops up: Select an existing key pair or create a new key pair
  - Choose 'Create a new key pair'
  - Key pair name: MyDLKeyPair
  - Click 'Download Key pair' which downloads the MyDLKeyPair.pem file.
  - Click Launch Instances
- Check the newly created instance under EC2 running instances

Step 2: Connect to DLAMI using SSH

The instructions below are for Windows. For other operating systems you can use the terminal program to connect.
The downloaded file from Step 1, MyDLKeyPair.pem needs to be converted to MyDLKeyPair.ppk to be usable by PuTTY to connect to EC2 instance

Converting .pem file to .ppk

- Start PuTTYgen
- Select RSA at the bottom of the displayed interface.
- Click Load and browse to the folder containing the .pem file. Change the file type to open from the default .ppk to 'All files'
- Select the .pem file downloaded in Task 1 and click 'Open'
- Click OK on the displayed message box PuTTYgen notice and click 'Save private key'
- Ignore the warning regarding passphrase by clicking 'Yes'
- Use the same name for the key as used for keypair and click Save
- Close PuTTYGen
- With the key in the correct format you should be able to connect to the EC2 instance created in Step 1

Connecting to the EC2 instance

- Start PuTTY
- Under 'Category' select 'Session'
- Enter ubuntu@Public-IP-Adress-Of-Instance
- In the 'Category' pane, expand 'Connection', expand SSH, and then click 'Auth'
- Click 'Browse' to select the .ppk file and click 'Open'
- Click 'Yes' to the message box PuTTY Security Alert
- A remote connection should open

Step 3: Start the MXNet Python 3 Environment

The details of running an application are in Deep Learning AMI – Developer Guide: https://docs.aws.amazon.com/dlami/latest/devguide/dlami-dg.pdf

- Each Conda command has the following pattern:

```
source activate framework_python-version
```

- You will test MXNet to see how easy it is

- Activate the MXNet virtual environment for Python 3

```
$ source activate mxnet_p36
```

- This activates the environment for MXNet with Python 3

Step 4: Test MXNet Code To test example network

- Change directory to cd /examples/keras-mxnet

- Run as
```
$ python cifar10_resnet.py
```

- There is no need to keep this program to run to completion and you should terminate it by pressing the key combination 'Ctrl-C'

- Close the SSH connection

The important thing to note is that the Deep Learning AMIs come pre-configured which makes it simple to run different machine learning models.

Task 6: Running a job on Elastic MapReduce (EMR)

Amazon EMR is a managed service that makes it easier to run big data frameworks such as Hadoop and Spark for data analysis. These frameworks along with open source projects such as Hive and Pig can be used to process big data for analytics.

In this task you will analyse the data from the log files using a Hive script which are both provided by AWS. The details are in links under 'Links' section below.

Step 1: Create an S3 bucket and a Key-Pair

An S3 bucket can be used to provide program script for running on the EMR cluster and also to store the results.

S3 bucket

- Navigate to S3 under 'Services' on AWS management console
- Name the bucket as 'myemrbucket' (note that the bucket name has to be globally unique)
- Keep all the defaults and click 'Create bucket'

key-pair

- Navigate to Key-Pairs under EC2 Dashboard, provide name as 'myemrkeypair' and select .pem or .ppk depending on which operating system you will use to connect to AWS
- Click 'Create key pair' and the file (.pem or .ppk) will be downloaded to your computer

Step 2: Create and launch a cluster

- Navigate to EMR under 'Services' in AWS management console
- Click 'Create cluster'
- An EMR cluster can be created using 'Quick Options' or 'Advanced Options'
- Click 'Go to advanced options'

Software Configuration

- Let the 'Release' pre-selected as emr-5.29.0
- Place check mark to select Hadoop 2.8.5, Pig 0.17.0, Hive 2.3.6, Hue 4.4.0, Spark 2.4.4
- The remaining steps are optional for 'Software Configuration'
- Click 'Next' to proceed to 'Hardware Configuration'

Hardware Configuration

- Let all the defaults settings selected
- Click 'Next' to go to 'General Cluster Settings'

General Cluster Settings

- Set myEMRCluster as the 'Cluster name'
- For logging, select 'myemrbucket' by clicking the folder icon
- Uncheck 'Termination protection'
- Click 'Next' to go to 'Security Options'

Security Options

- For EC2 key pair select 'myEMRKeyPair' created above (it might take some time to appear in the drop-down after its creation)
- Let all settings to be at default values and Click 'Create cluster'

It will take some time for the cluster to be created. The Cluster status would change from 'Starting –Configuring cluster software'. You should wait until the status changes to 'Waiting – cluster ready'.

Step 3: Connect to the cluster

- In the cluster 'Summary' tab observe one master and two core nodes running
- Under 'Security and access' click the 'Security groups for Master'
- EC2 Dashboard will open to display the 'Security Groups'
- Select 'ElasticMapReduce-master' under 'Group Name'
- Navigate to 'inbound' tab and click 'Edit'
- Click 'Add Rule' and for the new rule, select 'Type' as SSH and 'Source' as 'My IP' and click 'Save'

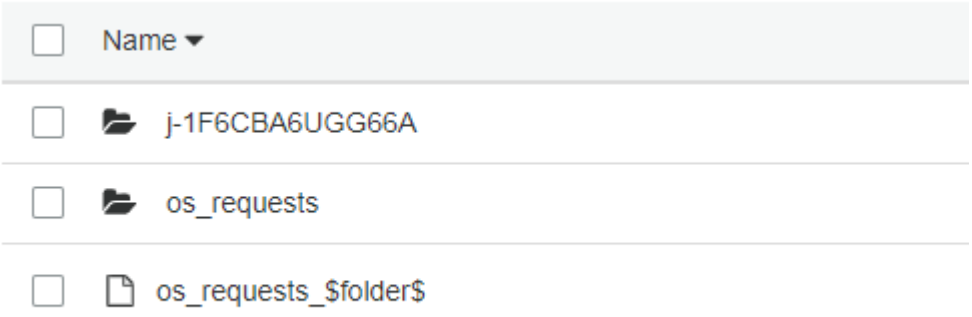
- From the summary tab on 'Amazon EMR' clusters link for 'myEMRCluster' copy the 'Master public DNS' and click SSH for details on connection details
- Open PuTTY (or another application for other operating systems) to connect to the cluster

Step 4: Run a Hive script on the cluster

- Navigate to 'myEMRCluster'
- Select 'Steps' and click 'Add step'
- In the 'Add step dialog
  - For 'Step Type' select 'Hive program'
  - For 'Name' enter 'MyHiveProgram'
  - For 'Script S3 Location' enter s3://*region*.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q
    Replacing region with the region name such as 'eu-west-2'
  - For 'Input S3 location' enter s3://*region*.elasticmapreduce.samples
    Replacing region with the region name such as 'eu-west-2'
  - For 'Output S3 Location' select the S£ bucket that you created 'myemrbucket'
  - Click 'Add'
  - The script will take some time to complete
  - You can view the results after the 'Status' has changed to 'Completed'

Step 5: Analyse the results

- Navigate to 'myemrbucket' where the results would have been written



- Select the folder os_requests and there would be a single file inside. Select the file and download it to view the results

- The file shows the number of access requests ordered by the operating system with the contents as:

```
Android 855
Linux 813
MacOS 852
OSX 799
Windows 883
iOS 794
```

Do not delete S3 bucket yet as you can use it in Task 7.

Task 7: Data Pipeline to export DynamoDB data to S3 bucket

In this task you will create a data pipeline to move data from a DynamoDB table to an S3 bucket. This creates an EMR cluster in the background and you must terminate it after use.

The data pipeline service requires use of an EC2 instance 'm3.xlarge' which is not available in 'eu-west-2' so you can create a DynamoDB table in 'eu-west-1' whereas the S3 bucket where you want to move the table's data may be in 'eu-west-2'. The description below assumes an S3 bucket named 'mybucket' and a DynamoDB table 'myDynamoDB'.

Step 1: Creating and activating a pipeline

- Search for 'Data Pipeline' service on the AWS management console.
- Click 'Get started'
- Click 'Create data pipeline'
- Enter name as 'MyDataPipeline'
- For 'Source' select 'Build using a template' and from the drop-down select 'Export DynamoDB table to S3'
- For 'Source DynamoDB table name' enter 'myDynamoDB'
- For 'Output S3 folder' enter 's3://mybucket/exports' where exports is a folder that will get created in 'mybucket'
- For 'Region of the DynamoDB table' enter 'eu-west-1' where you have created this table
- Under Schedule, 'You can run your pipeline once or specify a schedule' enter 'On pipeline activation'
- Under 'Pipeline Configuration' for Logging enter 'Enabled' and specify 'location for logs' as 'S3://mybucket/logs/'
- Under 'Security/Access' select 'Default' for IAM roles
- Click 'Activate'

- You may get a warning. You can select 'continue to activate the pipeline despite warnings'. The warning is about setting a time limit for an automatic termination of the EMR

Step 2: Monitoring and Testing

- The status would initially be displayed as 'WAITING_ON_DEPENDENCIES'
- After some time the status will be updated to 'WAITING FOR RUNNER' when the pipeline awaits for EMR cluster
- After some more time, the status will change to 'RUNNING'
- At this point if you navigate to EC2 running instances in 'eu-west-1' you will see that there are two EC2 instances created automatically by the EMR cluster for the pipeline
- The status will then change to 'FINISHED'

Step 3: Checking the exported data

- Navigate to S3 bucket and you should see two folder inside 'exports' and 'logs'
- There will be a text file under exports. Download and open it to find the contents of DynamoDB table

Task 8: Releasing the allocated resources

Make sure that you have released all the resources created in tasks above including:

- Delete the 'OrderFlowers' chatbot from Task 3
- Terminate EC2 instance and delete key-pair created in Task 5
- Delete S3 bucket, EMR cluster, EC2 instances and key-pair from Task 6
- Delete Security Groups, DynamoDB table, and Data Pipeline, terminate the EMR cluster from Task 7

Links

The links below are for your reference only in case further information is required to help complete tasks above:

Task 1

1. Using Amazon Rekognition
   https://docs.aws.amazon.com/rekognition/latest/dg/labels-detect-labels-image.html

Task 2

2. Amazon Textract and Amazon Polly
   https://docs.aws.amazon.com/polly/latest/dg/get-started-what-next.html
   https://github.com/aws-samples/amazon-textract-code-samples/blob/master/python/01-detect-text-local.py

Task 3

3. Sample chatbot using Amazon Lex
   https://docs.aws.amazon.com/lex/latest/dg/gs2-create-bot.html

Task 4

4. Build, Train, and Deploy a Machine Learning Model
   https://aws.amazon.com/getting-started/tutorials/build-train-deploy-machine-learning-model-sagemaker/
   https://aws.amazon.com/blogs/aws/sagemaker/

Task 5

5. Deep Learning AMI
   https://docs.aws.amazon.com/dlami/latest/devguide/dlami-dg.pdf
   https://docs.aws.amazon.com/dlami/latest/devguide/launch-config-test.html
   https://aws.amazon.com/getting-started/tutorials/get-started-dlami/?trk=gs_card
   https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/

Task 6

6. EMR
   https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-process-sample-data.html
   https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html

Task 7

7. Data Pipeline to export DynamoDB
   https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBPipeline.html#DataPipelineExportImport.Exporting
   https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBPipeline.html