Pablo Alcázar Morales
Diego Pedregal Hidalgo
Alberto Velasco Mata

Amazon SageMaker

– Machine Learning on AWS –

December 9, 2019

Springer Nature

Contents

Chapter 1
Introduction

Abstract Amazon SageMaker is a fully managed machine learning service. It allows you to launch a Jupyter notebook instance on the cloud, with just a few clicks, and make easy the duty of building and training learning models. You also can deploy models into a secure and scalable environment, and it is billed by minutes of usage.

1.1 Overview

Since SageMaker uses Jupyter notebooks, the programming language is Python, and uses his own Python framework, called sagemaker. But Amazon SageMaker provides tools to combine any other machine learning framework, like Apache Spark, TensorFlow, PyTorch and so on.

SageMaker is part of the amazon SDK that is written in Go language. The current version of Amazon SDK is v1.25.39 (last checked on 2019-11-20). The Amazon SDK allows the different amazon services to interact with each others on a simple way.

However, you can do much more than creating a notebook. With Amazon SageMaker you can create specialised computer instances and use them for our training Jobs. There are a few types of instances for different use cases. Also you can create algorithms, hyperparameter tuning jobs, create and train your models, etc.

But there is the most interesting point. As we said, Amazon SageMaker is also scalable. And the Amazon SDK allow us to interact with other Amazon services like EC2 instances, AWS Glue (that is a data analytics service), Amazon S3 (which is a bucket service for storage, where you can keep your datasets and graphics or models outputs that you may generate).

Chapter 2
Amazon SageMaker in detail

Abstract Through this chapter, we discuss the different functionalities and modules provided by Amazon SageMaker to perform all kind of data analysis by means of machine learning techniques.

2.1 How does Amazon SageMaker work

As we said before, the idea behind Amazon SageMaker is to provide clients with a fully managed service which allows them to integrate machine learning models into their own applications, in a more or less easy way.

2.1.1 Machine Learning models

To have a better understanding about this tool, it is important to review the typical workflow when creating a machine learning model:

1. **Generate example data**. This is probably the key part of machine learning. Obtaining good example data is essential to create a successful model. It is not only about finding good data, but also spending time exploring and preprocessing them (fetching, cleaning, transforming data are relevant operations).
2. **Train a model**. We can divide this task into two smaller ones:
    a. **Training the model**. Basically, training a model consists on choosing an algorithm (which should be intelligently chosen). For this purpose, Amazon SageMaker provides some built-in algorithms, which we'll discuss later.
    b. **Evaluating the model**. After training a model, it is required to evaluate its performance (accuracy) in order to know if this particular model is working as desired. Obviously, Amazon SageMaker provides the necessary tools for this task, too.

3. **Deploy the model**. After obtaining the model desired, the next step is about integrating it within your application. With Amazon, you no longer need to get involved in re-engineering or anything like that. You may simply deploy the model decoupling it from your application code, thanks to Amazon SageMaker hosting services.

It is obvious work does not finish here. Machine learning is a continuous cycle, so after deploying a model, you monitor its inferences, collecting what Amazon calls "ground truth", which could help your model to increase accuracy, repeating the process again and again.

2.2 Amazon SageMaker tools

Once the machine learning lifecycle has been clarified we are going to discuss the tools Amazon SageMaker offers to create models and integrate them into any application.

2.2.1 Preprocessing data

As we talked previously, before applying any kind of algorithm to a specific set of data, you should know how it is organised, its different fields and contents. This is known as exploring the data, for which you may use the Amazon SageMaker notebook instances, which basically are Jupyter notebooks (programming is done in Python), in which you can get to know more about your dataset.

Depending on the composition of this dataset, you probably need to transform it (preprocessing) to a format more efficient for training, from a computer's point of view. For this purpose, Amazon SageMaker recommends using either of these two methods:

a. Jupyter notebook instances.
b. Amazon SageMaker batch transform, that uses a model to transform your data automatically.

2.2.2 Training models

Amazon SageMaker creates training jobs to train models. To create a training job, you have to provide several information: where to store output, where is the input data stored, compute resources you want Amazon SageMaker to use, etc. But the most important one is the algorithm you want to use for training. Now, Amazon SageMaker offers several options:

a. **Built-In algorithms**. Some examples of them are K-Means, K-NN, PCA (we have discussed these algorithms during classes).
b. **Use Apache Spark**. Amazon SageMaker provides a library which allows you to use Apache Spark to train models, in a similar way Apache Spark MLLib does.
c. **Upload custom code**. If you prefer submit your own code, you are allowed to do so. Code must be written using Python and TensorFlow or Apache MXNet for model training.
d. **Use own custom algorithms**. You also can put your code as a Docker image and use it through a `CreateTrainingJob` API call.
e. **Subscribe an algorithm from AWS Marketplace**. Finally, if you are interested in using any algorithm which is already available at the Marketplace, you may pay for it.

As we can see, it is quite easy to start running some training for any given set of data you are willing to extract useful information from. Amazon SageMaker provides the client with a wide variety of possibilities. You only have to choose (although it is not the easiest task).

2.2.3 Model deployment

After training your model successfully, only one final step is left: the deployment of your model. Amazon SageMaker allow the clients to do this operation in two different ways:

a. **Amazon SageMaker hosting services**. If your intention is to get one prediction at a time, this is the best option.
b. **Amazon SageMaker batch transform**. However, if you are willing to obtain predictions for an entire dataset, this last option will be more efficient.

Summing up the chapter, we think Amazon SageMaker is a really interesting tool to make your Machine Learning projects succeed, offering a really straight forward way to do things, almost like "plug & play".

Chapter 3
Amazon SageMaker: Quick Start

Abstract Setting up Amazon Sagemaker is quite simple and can be done in a few minutes due to the integration with other *Amazon Web Services*.

3.1 Set Up

If you are familiar with *Amazon Web Services*, SageMaker set-up is quite simple. Once you have an AWS account, it is as easy as signing in to the Management Console and accessing to the Amazon SageMaker service.

3.2 Dataset containers

Amazon SageMaker uses *"buckets"* to store data. These are provided by another Amazon service: *Amazon Simple Storage Service (Amazon S3)*.

This way, it is mandatory to create an S3 bucket which will hold all the training data that we will use later. Multiple buckets can be made to keep datasets separated.

3.3 Create a notebook and start training

As seen in Section 2.2, Amazon SageMaker provides different possibilities to host our model and training processes. The simplest one is using a Jupyter notebook, but it requires an Amazon Notebook Instance to run it. It is just a cloud computing instance that manages and executes the Jupyter notebooks you create.

Once The Amazon Notebook Instance is created an a Jupyter notebook is running on it, you can start preprocessing the data and training your models on it. As mentioned in Section 2.2.2, Amazon SageMaker provides many algorithms in the

Python package `sagemaker`, which provides similar tools to the Scikit-Learn and Pandas libraries seen in class.

3.4 Conclusion

The main advantage of *Amazon SageMaker* is that it provides and hosts all you need, and it is on the cloud so you don't need the computational power neither the storage on your machine. It also has several tools to use and almost everything is integrated in the AWS cloud. However, we think this is also the main disadvantage, because despite being "simple" and having a lot of documentation, the knowledge of how Amazon Web Services works that you require to get into this platform is quite high. Having everything integrated in the same place is a trade-off: you need to know lots of small services to get all running.