Training ML models using GCP

Mick van Hulst

April 9, 2018

1 Introduction

This document describes a tutorial which can be used for connecting with GCP (Google Cloud Platform) and sending training jobs to Cloud ML. The folder with name 'templates' contains several templates. You can use these templates to train your models on Cloud ML.

Note: This tutorial assumes that you're using Keras to train your models. If you're using a different package, then you'll have to change the template 'train.py'.

2 GCP setup

As a preparation the TA's have created several containers[1] on GCP. The TA's have assigned a container to each group.

To work with GCP you'll need to install their SDK (use the following *tutorial*). This tutorial will prompt you to select a project for which you'll have to select 'Pilot project 1'.

3 Upload training data

To train your model, you first have to send your training data to GCP. Use the following (terminal) command to send your training data to your container:

```
gsutil cp -r path_to_data_file/data_file.extension \
gs://container_name/data_file.extension
```

[1] Basically a folder which can be assigned to specific users. This makes it possible to assign a container to each specific group. These folders/containers are used as a mechanism to save your data files and resulting models.

4 Folder structure and templates

Use the following folder structure for training your model on GCP:

```
root
├── trainer
│   ├── __init__.py
│   ├── train.py
│   └── cloudml-gpu.yaml
└── setup.py
```

Figure 1: Folder structure GCP

4.1 Use cases files

The file '__init__.py' is an empty initialization file.

The file 'train.py' is a template which can be used to combine your model with the structure needed to use GCP. The remainder of this tutorial assumes that you're using this template.

The file 'cloud-gpu.yaml' is a file which can be used to adjust the amount of GPU's you want to use.

The file 'setup.py' is a file that GCP uses to check which packages you want to use. Make sure to add all packages that you're using (excluding the standard Python packages) to this file.

4.2 Training your model

4.3 Local

Training your model locally with a small dataset is wise as it allows you to test your model after amending the changes which are described above. To test your model locally, use the following command:

```
gcloud ml-engine local train --module-name trainer.train \
--package-path path_to_root/trainer -- \
--train-file path_to_local_train_file/train_file.extension \
--job-dir ./tmp/test_script_gcp
```

4.4 Cloud

To test your model in the cloud use the following command, where the job name needs to be unique and where the region should correspond to the region of your container (region can be found at your container page on GCP):

```
gcloud ml-engine jobs submit training job_name --stream-logs \
--runtime-version 1.4 --job-dir gs://container_name/job_name \
--package-path path_to_root/trainer --module-name trainer.train \
--region region_name -- \
--train-files gs://container_name/data_file.extension
```

During training you can look into the progress of your model by navigating to the 'Logging' page at your GCP console. Make sure to apply a correct filter so that you only see logging data from the current job.

After training, you can use your model to make predictions. You'll have to download your model first and then make the predictions locally. Use the following command to download your model:

```
gsutil cp gs://container_name/job_name/model.h5 folder_to_download_to/
```