Training ML models using GCP

Mick van Hulst

March 30, 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 which this document will refer to and which can be used to train models on Cloud ML.

Note: This tutorial assumes that you're using Keras to train your models. This tutorial wasn't tested with any other packages.

2 GCP setup

As a preparation the TA's have created several containers[1] on GCP.

To work with GCP you'll need to install their SDK. Google offers a tutorial for this. During this tutorial you'll be prompted to select a project for which you need to select 'Pilot project 1'.

3 Upload training data

To train a model you first need to send your training data to GCP, you can use the following (terminal) command to achieve this:

```
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

To train your model on GCP the following folder structure is advised (see Figure 1).

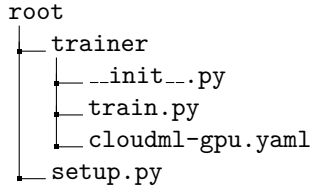

Figure 1: Folder structure GCP

The file '__init__.py' is an initialization file which needs to be created and can be left empty.

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've merged your model with this file.

The file 'cloud-gpu.yaml' is a file which can be used to adjust the amount of GPU's one wants 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 (excluding the standard Python packages) to this file.

4.1 Training your model

4.2 Local

Training your model locally with a small dataset is wise as it allows you to test if your model is working 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
```

4.3 Cloud

Now that you've tested your model locally and thus know that if a bug occurs it's not your model, you can safely train your model using GCP. Using a unique job name at every run and using the right region (which can be found at your

container page on GCP0, you can train your model on GCP using the following command:

```
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
```

After training your model you can use it to make predictions. You'll have to make these predictions locally and to download the resulting model you can use the following command:

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