Training ML models using GCP

Mick van Hulst   Dennis Verheijden

April 11, 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 (using 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. This can be done in a number of ways, most people will probably want to send a folder (with train, test and maybe validation data). This can be done using the following (terminal) command:

```
gsutil cp -r data_dir gs://container_name
```

---

[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.

Where `data_dir` is the local directory where the files are stored. You can also specify where the files have to be saved, by adjusting the destination url, for example to: `gs://container_name/data`.

For copying separate files you can remove the `-r` (recursive) parameter and specify file source and destination:

```
gsutil cp -r local_path_to_file/file.extension \
gs://container_name/file.extension
```

Downloading data can be done by switching the **gs://url** with the local folder.

4 Folder structure and templates

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

```
root
└── trainer
    ├── __init__.py
    ├── train.py
    └── config.yaml
└── setup.py
```
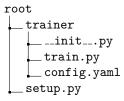
Figure 1: Folder structure GCP

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 `config.yaml` is a file which can be used to adjust the amount of GPU's you want to use during training.

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.

5 Training your model

5.1 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
```

5.2   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  --config path_to_root/trainer/config.yaml \
-- \
--train-file gs://container_name/data_file.extension
```

After the 'empty' parameter (`--`), you may add your own parameters that can be passed to the trainer.

During training you can look monitor 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 local_folder/
```