Take-home Final

Due Monday, May 13, 2024, 11:59pm

Problem 6.1

Use the CovidNYT database on the SciServer to perform the following statistical analysis from a Jupyter notebook:

(1) Using the Census data in the database, write and execute SQL queries to select the 200 Counties in the US with the highest and lowest population densities. Extract the total population in both categories. You can use the CasJobs of the SciServer to explore the databases interactively, and to debug your queries.

(2) The StatsC table contains the cumulative counts of both the Covid19 infections and deaths for each county in the US, for each day. Using the queries from the previous step, extract the maximum of the cumulative infections and deaths for the top and bottom 200 counties (in population density). This should give a total of 4 numbers (corresponding to Apr 18, 2021).

(3) Using the population counts, calculate the values normalized to a population of 100,000 and determine if there is a statistically significant difference between the highest and lowest population densities. Try to estimate the difference between the values in terms of standard deviations, both for the infection counts and for the deaths.

Hint: a notebook called HW10-sql-helper.ipynb has been placed into the shared directory. Run this in the regular class container.

Problem 6.2

The task is to classify a set of traffic signs from Germany. There are two binary files with the images in the shared data folder. The file signs1.bin contains RGB images of 12630 traffic signs, while signs2.bin has the gray scale version. The color images are 32x32x3 uint8, while the gray scale images are 32x32 uint8. There are altogether 43 distinct classes among the signs. There is a notebook called trafficsigns.ipynb in the shared folder that shows how to access the images. Primarily use the gray scale images in signs2.bin for this project. Play with the RGB if you have extra time.

Follow Chapter 17 on the Geron book to build an autoencoder to reproduce the images. Once done, use the lower layers of the autoencoder as a classifier. First run the example on the Fashion MNIST data, and then modify the code to accommodate the traffic signs.

In order to do this, you will need to split the sample into a training set and a validation set. Vary the size of the latent layer of the autoencoder to see how the categories found will depend on this parameter. Explain the results of the experiments.

You may want to save your model at the end, and then you can reload later. See

https://www.tensorflow.org/guide/keras/save_and_serialize

Setting up GPU containers

The second problem requires access to modern GPUs. The SciServer team has created a group called A100_users, that gives interactive access to a set of A100 GPUs. Each user will get a slice of an A100 GPU with 20GB GPU memory each. Each student in the class must have received an invitation to be part of this group. You should accept the invitation.

You should see the new domain in Compute's dropdown list called "A100 GPUs", and there is currently only a single GPU Essentials image available there that has Tensorflow and PyTorch preinstalled.

Now execute the following steps:

1. From the entry screen, go to Compute
2. Hit "Create container"
3. Enter a name, like "A100".
4. For the Domain, select "A100 GPUs".
5. The Compute Image is automatically set to GPU Essentials.
6. A Data Volume at the bottom of the page, named AS_171_205, should already be selected. If it is not, select it now.
7. Press "Create"
8. Open the Container

You will see the `/AS_171_205` as an additional resource. The files are in the `/data` subdirectory.

DO NOT OPEN ANY NOTEBOOK in this container, as this is shared, and you only have read privileges. Use this directory to copy/paste any of the example scripts to your working directory (e.g. your Finals or HW6 folder). You can list the samples directory from your working directory with:

```
import os
cpath = '/home/idies/workspace/AS_171_205/data/'
os.listdir(cpath)
```

You can read data files by prefixing the filename with cpath. Please test that your GPU scripts work with this image, and let me know if you need anything else. Try first `Chapter10.ipynb`.

These containers are fairly barebones. Some commonly used python packages may be missing. If this is the case, do not panic, you can install these yourselves.

To install additional python modules, like umap-learn, click on the small (+) sign next to the top of the tabs, open the Launcher tab, and select Terminal. In the linux command line in the terminal tab, type the install command e.g.

```
pip install umap-learn
```