Homework 4 - Part A

10-405/10-605: Machine Learning with Large Datasets

Due Wednesday, March 31st at 11:59 PM Eastern Time

Instructions: Submit your solutions via Gradescope, following the template below. Note that this assignment is different from previous ones; it does not contain a theoretical written part and the programming part is much more open-ended than before. Because of this, we will not be autograding your submission. Instead, you will submit a report consisting of responses to questions asked in the notebook, along with your underlying code. The report (and the underlying code) are worth 100% of your grade.

Submitting via Gradescope: When submitting on Gradescope, you must assign pages to each question correctly (it prompts you to do this after submitting your work). This significantly streamlines the grading process for the course staff. Failure to do this may result in a score of 0 for any questions that you didn't correctly assign pages to. It is also your responsibility to make sure that your scan/submission is legible so that we can grade it.

1 HW4 Part A: Machine Learning with the Million Song Dataset

1.1 Introduction

The goal of this assignment is to gain hands-on experience with training a machine learning model on a large dataset on cloud computing services.

In **Part A**, you will set up AWS (Amazon Web Services), access, convert the Million Song Dataset (MSD) into a usable format, and run Spark on AWS EMR. **START EARLY** because data conversion might take several hours.

Later in Part B, you will do EDA, data cleaning, and modeling in a Jupyter notebook and some (very limited) starter code. At the end, you will have a chance to optimize your pipeline and model.

1.2 Logistics

We provide the code template for Part A in *one* python script for converting the dataset and *one* Jupyter notebook for data loading and preparation. You should follow the instructions in the python script and notebook and implement the missing parts marked with '`# YOUR CODE HERE`'. **Unlike the previous homeworks, this homework gives you more freedom and less guidance so you may need to design your code from scratch and test it by yourself.**

Note that we will not autograde your code through Gradescope. Instead, you should submit your code along with your report (including plots, statistics, and short answers). Points will be given according to your answers in the report. We may also refer to your code submission as needed.

1.3 Getting lab files

You can obtain the starter code for **Part A**, `million_song_reader.py` and `HW4_partA.ipynb`, after downloading and unzipping `hw4-a.zip` from https://github.com/10605/released-hws/releases/tag/s21-hw4a.

1.4 Preparing for submission

Complete `million_song_reader.py` and `HW4_partA.ipynb`. Fill in corresponding written questions using this write-up `hw4-a.pdf` as a template.

1.5 Submission

1. First download your `million_song_reader.py` to your local computer. (You may use `scp <source> <destination>`). Then download the notebook to your local computer by going to `File -> Download as -> Notebook (.ipynb)`.
2. Submit your `million_song_reader.py`, `hw4.ipynb`, and this write-up `hw4.pdf` to Gradescope.

2 Part A: Data Conversion and Preparation

To get ready for Part B, you will need to set up your AWS account and redeem your $50 credits. You are also expected to access the MSD and transform it from `h5` format to `csv` while keeping only the features we need. Finally, you will set up Spark on an AWS EMR cluster and run some very preliminary analysis over the converted dataset.

Again, **PLAN TO START EARLY**. One reason is that the entire MSD is 280GB and running data conversion on it might take a few hours. It will be frustrating if your code runs for several hours before deadline then aborts due to a tiny bug and you have to re-run.

The other reason is that you may run into a few roadblocks when working with a large dataset on the cloud, especially if this is your first time performing such an exercise. Although we have included several tips in this writeup, you may still encounter some unexpected problems. Luckily, search engines are always your friend. If you have problems with something, someone elsewhere might have encountered the same problem! We recommend that you first try searching for solutions to your problems online (as this will likely be the fastest route to an answer), and if you're still stuck to then post on Piazza and / or come to office hours.

2.1 Setting up AWS

Amazon Web Service (AWS) is one of the most comprehensive cloud computing platforms. Although there are many other options for cloud computing (e.g. Azure, GCP), you should use AWS for this particular assignment. This makes us supporting the entire class much easier.

2.1.1 Create AWS account & redeem credits

Create an AWS account if you don't already have one. You will receive an email from the course staff containing a $50 coupon code on AWS. You can redeem it at `Billing Dashboard -> Credits`. **Note that this $50 coupon is meant to cover your costs for ALL OF HOMEWORK 4 (i.e., for both Parts A and B**).

2.1.2 Cost management

Make sure you manage your cost while doing this assignment. To do this effectively, we highly recommend you learn about EC2 pricing and S3 pricing, and **create an AWS Budget of 35 dollars so that you are alerted when you are close to your budget limit.** Technically, the course staff will not be responsible for covering extra charges incurred beyond the supplied coupon.

2.1.3 AWS S3

You have to create a S3 bucket in order to store your converted MSD into it.

2.1.4 IAM Role

To have read/write access of S3, you also have to create a role with the `AmazonS3FullAccess` policy attached under `IAM -> Roles -> Create role` (here).

2.1.5 Configure and launch EC2

Configure and launch EC2 instances to develop and run your data conversion. Here are some guidelines (note: we will cover this process in detail in lecture on Wednesday 3/17):

- Choose the default image (i.e. something like `Amazon Linux 2 AMI (HVM)`). Then choose `t2.micro` or `t2.medium`. Note that `t2.micro` can be used for development, but might not have enough memory to process your entire dataset.

- Select a subnet starting with **"us-east-1"**. Remember your choice since you have to select the same subnet for the volume you are going to create.
- Choose the IAM role you just created.
- Keep options default and move on. You will need to select a security group (this can be changed after creating the instance). You may select the "default" security group and add an inbound rule to it for the `SSH` service with `source` set to be `Anywhere`.
- Create a key-pair if you haven't already.

The image might not come pre-installed with Python 3 so you could do so via `$ sudo yum install python37`. You should also install packages you import in the python file. Use `pip3` with the `--user` flag to avoid `sudo`.

Again, cost management is key. You could launch a single `t2.micro` while you are developing to save cost. You should launch up to two `t2.medium` instances while you run the actual conversion. You should stop (**not** terminate, see distinction here) your instances when you are not using them for a while. When an instance is stopped, you will only be charged EBS storage fees of your data but not instance running fees. You should terminate them **after you have downloaded your code** when you are done.

2.1.6 Create MSD volume

Create a volume from the AWS MSD snapshot. Select the same "subnet" as the EC2 instance's, (an example subnet might be "us-east-1d"). **Remember the subnet must start with "us-east-1", or the snapshot of MSD cannot be found**. Attach and mount the MSD volume to the EC2 instance. We will run through this process in more details in lecture. Remember to delete this volume after you are done with this homework.

2.1.7 Credentials

You may not need to do this in this homework. But if you want to use the AWS command line tool, you may have to configure credential files in order to let your program access AWS services. The easiest way would be to run `$ aws configure`. See Configuration and credential file settings for details.

2.1.8 Development on AWS

You could use whatever IDE/editor you are comfortable with. For starters with remote development, you could try Visual Studio Code with the Remote-SSH extension (see details here) or Vim.

2.2 Million Song Reader

After completing the above steps, we are ready to operate on the dataset and convert it into a desirable format. The starter code is provided in `million_song_reader.py`. Upload this file to your instance. (You may use `scp <source> <destination>`)

General guidance and tips of converting the data are the following. Please refer to the comments in the `.py` file for detailed instructions. You don't necessarily have to follow our code template if you think it's easier to write your conversion script from scratch.

- You should start by understanding how to access the fields in `h5` format (instructions are given in the comments under the function `process_h5_file`).
- For you to know what features we are extracting from the original dataset, the complete list of fields is provided *at the top* in the python file. We want to include these fields in the resulting `csv` files.

  Take a look at MSD's full field list. Briefly explain why we include this specific subset of fields but not the others. (Hint: think about what category of features we omit and why that might be a sensible choice.) *[15 points]*

  We include this specific subset of fields because it contains mainly metadata and some features are in the analysis category of features, but a lot of the analysis features has been excluded. This might be a sensible choice because we are only interested in preliminary analysis over our converted dataset later, though it's somewhat a subjective choice.

- You should discard all records with `song_hotttnesss` field being `NaN`. (After all, this is what we are predicting for!)
- The dataset volume is attached to the instance and therefore can be access "locally" (we can treat it as local) from the instance. Then we need to generate temporary `csv` files in batches and upload them to S3.
- To speed up the process, you may want to split the jobs into different proportions and distribute them to different threads or different machines to allow parallel computation.

  The starter code for setting up the command-line arguments is provided. After you finished, you will be able to run `python million_song_reader.py 4 0`, which means split the job into 4 parts and this machine will work on job number 0. You can run this code on a single machine (multi-thread) or multiple machines to speed up the conversion.

  Again, the specific parameters are up to you. From our experience, running conversion to saturate performance on two `t2.medium` instances will do the job within 2 hours. You should experiment with the parameters while monitoring machine resource utilization (e.g. CPU utilization by `top` or `htop` and disk I/O by `iostat`) to optimize for cost.

Briefly describe your final configuration and the approaches you took to arrive at it. *[20 points]*

The configuration I used was a multiple thread configuration that used 2 instances with 2 cores. The processing was split with worker 0 processing alphabets A,C,E,... and so on. The approaches I took included other configurations such as one without multiprocessing. While that is easier to implement, it would have taken up to half a day to process which could be problematic especially if there were errors. Therefore it got me to the approach where multiprocessing was necessary.

Report the time and cost the conversion step took in total. *[10 points]*

~1.5 hr time and ~$8

Briefly describe any pitfalls or obstacles you faced during the data conversion process and how you overcame them. *[10 points]*

I had issues with worker groups only uploading 1 or 2 files and the try statement for the wrapper function fail, which was caused by a buggy process function. I also had issues reading the h5 files which was fixed by using os.path.join inside my os.walk loop.

2.3 EMR and Spark

With our data ready in S3, it's now time to configure and create an EMR (Elastic MapReduce) cluster and run Spark, starting from the notebook `HW4_partA.ipynb`. Include at least Hadoop, JupyterHub, and Spark in your cluster software configuration. Set `maximizeResourceAllocation` to true (see here) in software settings. Select your EC2 key pair.

You'll have to do ssh port forwarding (recommended) or attach additional security groups/inbound rules to the master node in order to access the JupyterHub web interface. This step could be tricky and we will cover how to do this in detail in lecture. Then, log in to jupyter (learn about login credentials here), upload your notebook, and start working!

Again, cost management is key. Because we might be running a cluster of machines, this could easily blow up your budget. We recommend using at most 1 Driver and 1 Core of type `m5.xlarge` while developing and debugging on a subset of MSD. You could scale this up to multiple Core workers when doing the final run. You may also utilize AWS spot instances to save cost during development.

Note that, although EMR is made up of EC2 instances, unlike EC2, you cannot stop an EMR cluster - you can only terminate it. See here for why this is the case. You should plan your strategy accordingly. **Do not forget to download your code** before you terminate a cluster when you are done.

2.4 Notebook: Data Loading and Preparation

(a) *[15 points]* List the 19 numeric features:

1. artist familarity (float)
2. artist hotttness (float)
3. artist latitude (float)
4. artist longitude (float)
5. artist terms freq (array float)
6. artist terms weight (array float)
7. danceability (float)
8. duration (float)
9. end of fade in(float)
10. energy (float)
11. key (int)
12. key confidence(float)
13. mode (int)
14. mode confidence (float)
15. start of fade out (float)
16. tempo (float)
17. time signature (int)
18. time signature confidence
19. year (int)

(b) *[30 points]* Include the output of the final cell (make sure you run on the entire dataset!):

42

3 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. (a) Did you give any help whatsoever to anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. (a) Did you find or come across code that implements any part of this assignment?

   (b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).