Rodolphe Barlogis

Application Task Assignment

September 8, 2019

The following report account to provide a clear and fidel overview of the job done for the given task assignment.

1 Database design

[!h]

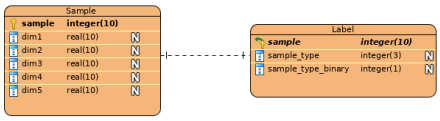

Figure 1: Designed with Visual Paradigm

We keep in mind that we need to avoid redundancy. Here sample is the obvious primary key.
We use datatype that will occupy too much place on disk real type has a range of 1E-37 to 1E+37 it is maybe already too much For the label we have, we use integer and not string ; it is easier to manage int than and it takes less space on disk

For this toy project, the design we have chosen does not make a lot of sense except for best cognitive aprehension of the data by separating dimensions and labels. Eventhough, some label could maybe be used as variable for other label

2 Deployment & ETL

A simple makefile has been written

```
PASSW = test
HOS = localhost
USE = postgres


runcont:
	sudo docker run --rm --name pg-docker -e POSTGRES_PASSWORD=$(PASSW) -d -p 5432:5432 postgres
	sleep 10
	PGPASSWORD=$(PASSW) psql -h $(HOS) -U $(USE) -c "CREATE DATABASE dbtoy;"
	PGPASSWORD=$(PASSW) psql -h $(HOS) -U $(USE) -d dbtoy -c "CREATE TABLE sample (sample_num
integer PRIMARY KEY,dim_1 real,dim_2 real,dim_3 real,dim_4 real,dim_5 real);"
	PGPASSWORD=$(PASSW) psql -h $(HOS) -U $(USE) -d dbtoy -c "CREATE TABLE label (sample_num
integer REFERENCES sample (sample_num),sample_type integer,sample_type_binary integer,PRIMARY KEY
(sample_num));"

	python3 dataConn.py
```

it is presumed that (the latest release of) postgreSQL is already pulled and it is written as if we do not want to persist data generated (we could add **mkdir -p $HOME/docker/volumes/postgres**)

The script **dataConn.py** perform the ETL. Given the small amount of data we could load everything on RAM. We did not check if the data was sane. Also, we did not care about python float approximation

3 API design

4 Data visualization

Lost a litle bit of time trying to set up something with java, but eventually went back to cherished python.

I must admit the result is hideous, It took a lot of time trying more complex stuff that ended up not working.

5 Database partitioning

We could naively dispatch the data. we would probably get a satisfying distribution in each node if the number of instances for each is sufficient

we could choose to first and then add fake data with common algorithms to handle unbalanced labels like **SMOTE()** to reproduce the distribution. (Maybe not a good solution).

Eventually, two solution have been implemented.

also make a grid, but the complexity could explose with label dimension

We stick to a simple euclidan geometry to calculate distance and so But you got the spirit Problem her is that label are not conituous

You can see below an illustration with one dimension density

The code is not general. not really good. There are some problem at the limits ; for example if we want to make a lot of partition with not a lot of instances (in reality there is no reason to do it) . for industrial use, it would need to be enhanced.

We could reduce approximation by blocking one (or more) while calculating density for others. And dispatch it the same way. For huge amount of data and a lot of label, I see this method as the best among the one proposed

6 Horizontal scaling