

UC Davis STA 242 2015 Spring Assignment 5 [1]

Wenhao Wu, 998587583

May 30, 2015

1 Algorithm Design

1.1 Compute the Deciles

In order to compute the deciles of the total fare less the tolls, denoted as f_{net} , we count the occurrence of each value of f_{net} . The benefits are:

- There are much more records than the possible values of f_{net} in the original data. Consequently, it **saves tremendous memory usage** by counting the occurrence.
- This algorithm is highly **compatible with parallel processing**. We can keep multiple tables to count the occurrence of each value of f_{net} for different data files, update these tables fully in parallel, and then merge these tables to compute the deciles.

In both of our implementations, we build a table to count the occurrence for each pair of data files by updating it sequentially as we read in a new piece/bulk of record(s), then combine the 12 tables to compute the deciles.

1.2 Solve The Linear Regression

Denote the trip time as t and the surcharge as f_s , respectively. In the two regression tasks, the responses are denoted as \mathbf{y} , a n -by-1 vector of f_{net} in all records. In the first regression tasks, the predictors are denoted as \mathbf{X}_1 , a n -by-2 matrix where the first column represents the t from all records and the second column is an all 1 vector. In the second regression tasks, the predictors are denoted as \mathbf{X}_2 , a n -by-3 matrix where the first and the second columns represent the t , and f_s from all records and the third column is an all 1 vector. Theoretically, the coefficients of the linear model can be computed as

$$\beta_i = (\mathbf{X}_i^H \mathbf{X}_i)^{-1} \mathbf{X}_i^H \mathbf{y}, i = 1, 2. \quad (1)$$

Apparently, the sufficient statistic for the linear regression tasks are $\mathbf{X}_i^H \mathbf{X}_i$ and $\mathbf{X}_i^H \mathbf{y}$, $i = 1, 2$ which has very low dimension. Moreover, these sufficient statistics can be updated sequentially as we read in a new piece/bulk of record(s), and are again highly compatible with parallel processing.

In both of our implementations, we update $\mathbf{X}_i^H \mathbf{X}_i$ and $\mathbf{X}_i^H \mathbf{y}$, $i = 1, 2$ sequentially for each pair of data files, then combine the 12 set of statistics by summing them up and solve the linear problem as in (1) to get the coefficients for the regression models.

2 Data Inspection, Pre-Processing and Extraction

Due to the limited hard drive space available on my workstation, I keep the original .zip files without decompressing them. Firstly we check that the “data” and “fare” files match each other row by row in the 3 index fields “medallion”, “hack_license” and “pickup_datetime”. To do so, we primarily make use of a combination of shell commands `unzip`, `cut`, `diff`, IO redirection and pipe commands to compare the 3 fields in each pair of files. (See `checkmatch.sh` in the Appendices.) We verified that the files indeed match in pairs.

During the inspection, we also notice that “trip_fare.8.csv.zip”, “trip_data.9.csv.zip” and “trip_fare.9.csv.zip” contains duplicated .csv files, which are removed manually.

In both of our implementations, we build a “connection” to read in the output of shell pipe commands to extract the data. The shell command to extract “surcharge”, “tolls_amount” and “total_amount” from the “fare” files is

```
unzip -cq ../data/trip_fare_n.csv.zip | cut -d , -f 7,10,11
```

According to the data file description [], roughly 7.5% of all trips’ “trip_time” is wrong so we take a safe approach to extract “pickup_datetime” and “dropoff_datetime” from the “data” files. The corresponding shell command is

```
unzip -cq ../data/trip_data_n.csv.zip | cut -d , -f 6,7
```

Later we take the differences between them as the actual trip time. Fortunately, both these two fields have a very neat format as “%Y-%m-%d %H:%M:%S” which can be easily processed.

3 Implementation in Python

Our first implementation is based on Python3. The parallel processing is implemented with package “multiprocessing”: we define a worker function `analyze_file()` to compute the count of occurrence table and the sufficient statistics for the two linear regression tasks for a single pair of data/fare files. A total of 12 copies of this worker function are mapped to a pool of multiple processes and run in parallel. The results are then combined, from which the deciles are computed and the 2 linear regression problems are solved.

The worker function `analyze_file()` has a coroutine structure []: it is mainly composed of a “source” function `parse_file()` which read in one line from a pair of data/fare files, process it, and send the result to a “sink” function `accumulate_lines()`, which is in charge of updating the count of occurrence table for the total amount less the toll and the sufficient statistics for the regressions.

In terms of data structure, the count of occurrence table is updated as a python `dict` object and later converted to a pandas `Series` object to enable easy combination. The sufficient statistics are represented as numpy `ndarray` objects.

4 Implementation in R

5 Results

5.1 Deciles and Regression Results

5.2 Running Time Comparison

6 Conclusion

References

- [1] Wenhao Wu. STA 242 Assignment 5: Working with “Big Data”. [git@bitbucket.org:shasqua/stat242_2015_assignment5.git](https://github.com/shasqua/stat242_2015_assignment5.git), 2015. [Online; accessed 30-May-2015].

Appendix: Source Files