Exercise set 4

Hou, Jue

014695647

* Exercise 1

|  |  |
| --- | --- |
| GPS Coordinate | Country |
| 60.2576009,24.9345427 | Finland |
| 40.7690327,-73.983803 | United States of America |
| 39.9163488,116.3949606 | China |
| 51.4955324,-0.1407132 | United Kingdom |
| 60.2039806,24.963352 | Finland |
| -25.3456376,131.0283696 | Australia |

* Exercise 2

For all three different regression approaches, I choose 100 as the number of iterations with 1e-4 as step size. It seems that a gradient descent algorithm is used to fit the training data. Therefore if step size is too big, we can easily get infinity for weight vector. If number of iterations is too small, we will not get convergence. I once tried the configuration as what is showed in the document of Spark, which is to iterate 100 times with 0.5 step size, and it turns out to be infinite weight vector. I also tried to iterate 100 times with 1e-10 step size and it almost returns zero vectors, which means it did not get enough iterations but somehow get convergence. So, I choose 100 as the number of iterations with 1e-4 as step size to ensure reasonable weight vectors.

The result can be showed as follow:

|  |  |  |
| --- | --- | --- |
| Model | Weight Vector | Test Error(MSE) |
| Linear Regression With SGD | [4.059e-7, 2.017e-5, -3.394e-5, 1.704e-5] | 4.763e-5 |
| Ridge Regression With SGD | [4.059e-7, 2.017e-5, -3.394e-5, 1.704e-5] | 4.763e-5 |
| Lasso Regression With SGD | [0, 1.966e-5, -3.291e-5, 1.564e-5] | 4.787e-5 |

The performance of the algorithms is actually better than I expected, since the number of iteration has been bounded. And I also notice that the algorithm will stop iteration once convergence gets detected, which means that it sometimes won’t iterate for 100 times. However, because of manually setting parameter, it requires more experiment with different setting to avoid some suspicious results for example almost zero weight vector. In my opinion, we can instead implement adaptive step size algorithm with unbounded iteration but with early stopping strategy. If it is for linear regression, Newton’s method is also an option, since it will analytically solve the problem and reach convergence in only one step.

* Exercise 5
  + HDFS and Tachyon are essentially both distributed file system and they share some similar feature of distributed system, for example they both have the structure of master and workers. Their masters and workers are responsible for similar task. They are both infrastructure for Spark and can both provide file API to Spark.

HDFS and Tachyon is implemented with different design principle. The goal of HDFS is to be reliable. However, the goal of Tachyon is to improve the throughput performance and eliminate the bottleneck of writing and reading. Therefore, their implementations are very different and Tachyon outperforms in-memory HDFS by 110x for writes. HDFS will create multiple replications in order to be reliable. But Tachyon will minimize replication as much as possible. HDFS can interact with hard drive and provide file API. Tachyon is an in-memory file system built on replication based storage systems, such as HDFS. Tachyon has Lineage as part of strategy of fault-tolerance, but HDFS do not have. Tachyon implements hierarchical storage to ensure the performance, while HDFS did not. Tachyon enables data in-memory sharing so that Spark can read same data for different task, but HDFS cannot.

* + Spark streaming and dataflow model are both programming models, which are designed to process unbounded data or streaming. And they both can receive from different data source. There of cause are some differences in API implementation. But they both have similar operation like transformation: map and reduce for Spark streaming and apply for dataflow model. And internally, Spark streaming will process input data as discretized streams, a sequence of RDDs. Meanwhile dataflow model operate on pipeline, a data processing job. Spark streaming and dataflow model both have window operation. And they can both operate on window. But dataflow model allows event-time ordered results. It is possible to window data into different type of windows. However, the window of Spark streaming seems to be fixed. Also, dataflow model has a different windowing model and some other feature if compared to Spark streaming. Since the design principle of dataflow model is to fit the requirement of real world, the code of dataflow model is more clear and simple than Spark streaming. In addition, dataflow provides the possibility to upgrade streaming job during the data transmission, which means upgrading without losing existed windowed states.