INF344 TP2: Matrix-vector Multiplication

May 26, 2014

In this exercise, we will write a MapReduce code that given a $n \times m$ matrix A and a vector v with m elements, it will compute the product $w = A \cdot v$, where w is defined as:

$$w_i = \sum_{j=1}^{m} A_{ij} * v_j \; i = 1, \ldots, n.$$

A short description of how to implement this efficiently in MapReduce follows. There are two MapReduce jobs:

1. in the first job we multiply v_j with every element in the jth column of A, for every j. We obtain a new matrix B, where $B_{ij} = A_{ij} * v_j$ for all i, j.

2. in the second job we compute the sum $\sum_j B_{ij}$ for each row i so to compute v_i.

You should write map and reduce functions for each of the two jobs. A and v are stored in two separated files in your input directory. They are stored in a *sparse* form, that is, a set of lines "i j A_{ij}" and "j v_j", respectively, where i denotes a row and j a column of A and $A_{ij} \neq 0$; i, j, A_{ij} and v_j are separated by a space character.

Remember that A and v will be split in several parts, with each mapper receiving either a line of A or a line of v. To tell them apart you can use the fact that each line contains three integers in the case of the matrix or two in the case of the vector. Remember that in the reduce function for each key you get a stream of data which is stored in an Iterator in Java. It is not possible to reset a stream of data (and an iterator) nor to predict the order

by which elements are sorted. So you might need to store something in main memory. For this task, you can assume that one single column of the matrix fits in main memory.

For efficiency issues, you should use a combiner. A combine function is executed in a mapper node and before reduce tasks. It is used to minimize the amount of data sent to reducers (e.g. (jaguar,2) rather than (jaguar,1)(jaguar,1) when counting the number of words). The input of a combiner consists of the output of a map task. There is no guarantee that the combiner will be executed,this should be taken into account when writing the combine function.

You should submit your project by **June 3rd, 1pm**. Everything sent after that date will **not** be considered.

1 More Info

A skeleton project is available for download on the course Web site.

This project contains the following JAVA files :

MatrixVectorMult.java This the main file and the only one that you need to modify. A submission on the submission server will contain only this file. The TO DO annotations point the parts that need to be filled in.

PublicTests.java Contains tests that will be run on your code.

BaseTests.java A superclass of PublicTests with helping methods.

In the project you will also find a *data/* directory. Before running your program this will contain only the folder inputDirectory, where you find a file *matrix.txt* and a file *vector.txt*. It is recommended to test first using the content already existing in these files, as PublicTests is configured accordingly.

After the execution of the program, the output is found in the folder *data/resultDirectory*. You can also consult *data/intermediaryDirectory*, which contains the intermediary results of the first job.

Structure of MatrixVectorMult.java and what to implement

You are required to implement the mappers and reducers for 2 jobs.

class FirstMap is the first Mapper class to implement. As it can read either from the matrix file or from the vector file, you need to provide implementation for both cases.

class FirstReduce is the first Reducer class to implement. You need to take into account it is not possible to iterate twice.

class SecondMap is the second Mapper class to implement. It is the implementation of an identity function.

class SecondReduce is the second Reducer class to implement. It should perform the necessary steps for the completion of the algorithm.

class CombinerForSecondMap is the Combiner class for the second map function. The combiner is used for optimization purposes.