Big Data Analysis

Practical 7

Objective :

Implement any one of the analytic algorithm using mapreduce by handling larger datasets in main memory.

• PCY/Multi-Hash/SON algorithm

• Regression

• K-means Clustering

Roll No. & Name : 18bce183 & Prince Prajapati

Submitted to: Prof. Jaiprakash Verma

Regression in MapReduce

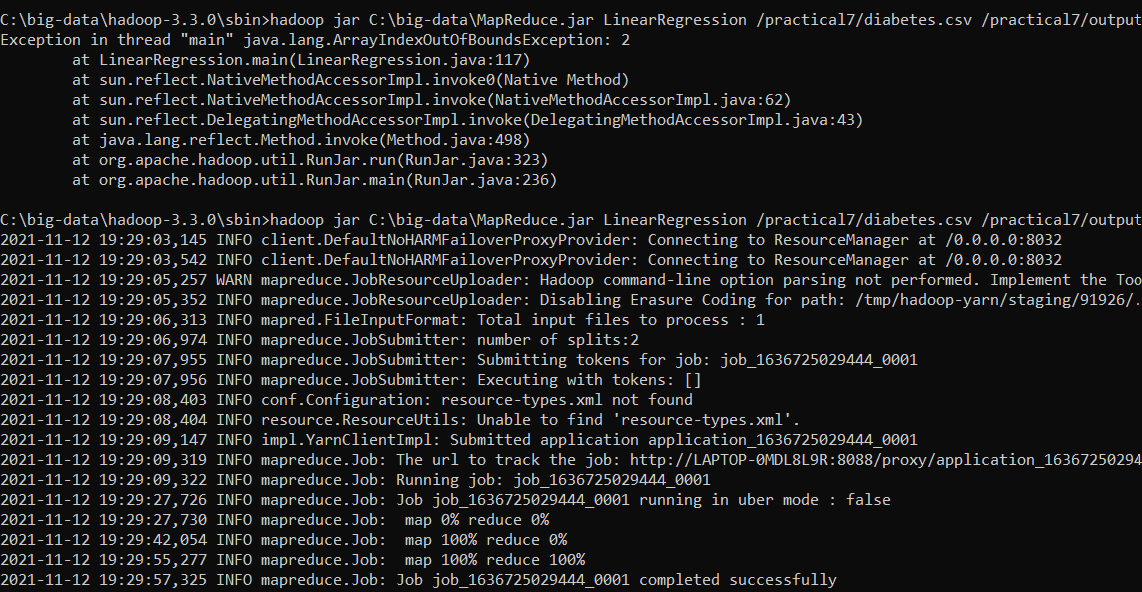
Sometimes, with big data, matrices are too big to handle, and it is possible to use tricks to numerically still do the map. [Map-Reduce](https://en.wikipedia.org/wiki/MapReduce) is one of those. With several cores, it is possible to split the problem, to map on each machine, and then to aggregate it back at the end.

**Code :**

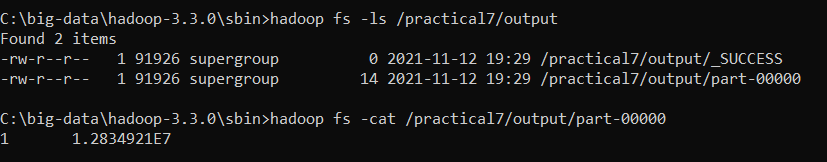
|  |
| --- |
| import java.io.BufferedInputStream;  import java.io.BufferedReader;  import java.io.DataInputStream;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  import java.io.InputStreamReader;  import java.net.URI;  import java.net.URISyntaxException;  import java.util.ArrayList;  import java.util.Iterator;  import java.util.List;  import org.apache.hadoop.filecache.DistributedCache;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.\*;  import org.apache.hadoop.mapred.\*;  public class LinearRegression {  public static class LinearRegressionMapper extends MapReduceBase implements Mapper<LongWritable, Text, LongWritable, FloatWritable>  {  @Override  public void map(LongWritable key, Text value, OutputCollector<LongWritable, FloatWritable> output,  Reporter reporter) throws IOException {    /\*  J(theta) = sum((y\_predict(x)-y)^2)  y\_predict(x) = theta(0)\*x(0) + .... + theta(i)\*x(i)  \*/    String line = value.toString();  String[] features = line.split(",");    List<Float> values = new ArrayList<Float>();    // read the values and convert them to floats  for(int i = 0; i<features.length; i++)  {  values.add(new Float(features[i]));  }    // calculate the costs  output.collect(new LongWritable(1), new FloatWritable(costs(values)));    }    private final float costs(List<Float> values)  {    float costs = 0;  //all right we have all the theta values, lets convert them to floats  String[] theta = {"0.01","0.03", "4", "2", "0.9", "2", "0.8", "0.9", "2", "3", "0.1"};    //first value is the y value  float y = values.get(0);    // Calculate the costs for each record in values  for(int j = 0; j < values.size(); j++)  {  //bias calculation  if(j == 0)  costs += (new Float(theta[j]))\*1;  else  costs += (new Float(theta[j]))\*values.get(j);  }    costs = (costs -y)\*(costs - y);  return costs;    }    }      public static class LinearRegressionReducer extends MapReduceBase implements Reducer<LongWritable, FloatWritable, LongWritable, FloatWritable>  {  @Override  public void reduce(LongWritable key, Iterator<FloatWritable> value,OutputCollector<LongWritable, FloatWritable> output, Reporter reporter)  throws IOException {      //The reducer just has to sum all the values for a given key      float sum = 0;    while(value.hasNext())  {  sum += value.next().get();  }    output.collect(key, new FloatWritable(sum));  }    }  public static void main(String[] args) {    JobConf conf = new JobConf(LinearRegression.class);    conf.setJobName("linearregression");      conf.setOutputKeyClass(LongWritable.class);  conf.setOutputValueClass(FloatWritable.class);    conf.setMapperClass(LinearRegressionMapper.class);  conf.setCombinerClass(LinearRegressionReducer.class);  conf.setReducerClass(LinearRegressionReducer.class);  conf.setInputFormat(TextInputFormat.class);  conf.setOutputFormat(TextOutputFormat.class);    FileInputFormat.setInputPaths(conf, new Path(args[0]));  FileOutputFormat.setOutputPath(conf, new Path(args[1]));  try {  JobClient.runJob(conf);  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Output:**

**hadoop jar C:\big-data\MapReduce.jar LinearRegression /practical7/diabetes.csv /practical7/output**

****

**Output showing square-error of regresion**

****

**Conclusion:**

After implementing this practical now I have clear understanding about regression in mapReduce and how it works with mapReduce.