BDA

Practical 7

19BCE248

AIM: Implement any one of the analytic algorithms using mapreduce by handling larger datasets in main memory. PCY/Multi-Hash/SON algorithm Regression K-means Clustering.

CODE:

import java.io.\*;

import java.util.\*;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Prac7 {

public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{

private Text word = new Text();

private Text center = new Text();

List<Integer> Centroids;

Path centroids;

FileSystem fs;

protected void setup(Context context) throws IOException, InterruptedException{

Centroids = new ArrayList<>();

centroids=new Path("hdfs:/prac7/centroids.txt");//Location of file in HDFS

fs = FileSystem.get(context.getConfiguration());

BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(centroids)));

String line;

line=br.readLine();

while (line != null){

Centroids.add(Integer.parseInt(line));

line=br.readLine();

}

}

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

int datapoint = Integer.parseInt(word.toString());

int min = Integer.MAX\_VALUE;

for(int i=0;i<Centroids.size();i++) {

int dist = Math.abs(datapoint-Centroids.get(i));

if(dist<min) {

min=dist;

center.set(Centroids.get(i)+"");

}

}

context.write(center, new IntWritable(datapoint));

}

}

}

public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

List<Integer> newCentroids;

Path centroids;

FileSystem fs;

protected void setup(Context context) throws IOException,InterruptedException{

newCentroids = new ArrayList<>();

centroids=new Path("hdfs:/prac7/centroids.txt");//Location of file in HDFS

fs = FileSystem.get(context.getConfiguration());

}

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum = 0;

int length = 0;

for (IntWritable val : values){

sum += val.get();

length++;

}

result.set(sum/length);

newCentroids.add(Integer.parseInt(result.toString()));

}

protected void cleanup(Context context) throws IOException,InterruptedException{

FSDataOutputStream out = fs.create(centroids, true);

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(out));

for(Integer itr: newCentroids) {

System.out.println(itr);;

bw.write(itr+"\n");

}

bw.close();

}

}

public static void main(String[] args) throws Exception {

for(int i=0;i<3;i++) {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "KMeans");

job.setJarByClass(Prac7.class);

job.setMapperClass(TokenizerMapper.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileSystem fs = FileSystem.get(conf);

if(fs.exists(new Path(args[1]+"/"+i))){

fs.delete(new Path(args[1]+"/"+i), true);

}

FileOutputFormat.setOutputPath(job, new Path(args[1]+"/"+i));

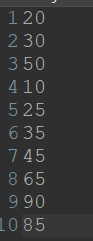
job.waitForCompletion(true);

}

}

}

INPUT:

OUTPUT FILE:

