|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
|  | Name of the student: | | Yash Sankpal | | | | Roll No. | | 8695 | |  |
| Practical Number: | | 6 | | | | Date of Practical: | |  | |
| Relevant CO’s | | At the end of the course students will be able to apply appropriate algorithms for extracting knowledge from given dataset. | | | | | | | |
| Sign here to indicate that you have read all the relevant material provided Sign:  before attempting this practical Practical grading using Rubrics    Indicator Very Poor Poor Average Good Excellent | | | | | | | | | |
|  | Timeline  (2) | Practical not submitted (0) | | More than two session late (0.5) | Two sessions  late (1) | One session late (1.5) | | Early or on time (2) | |  | |
| Code de- sign (3) | N/A | | Very poor code de- sign(0) | poor design  (1) | design with good coding  standards  (2) | | Accurate  Design with  better  coding standards(3) | |
| Execution  (3) | N/A | | Very less execution (0) | little execu-  tion.(1) | Major execu- tion(2) | | Entire code execution (3) | |
| Postlab (2) | Both an- swers wrong(0) | | N/A | One answer  correct (1) | N/A | | Both an- swers correct  (2) | |
| |  |  | | --- | --- | | Total Marks (10) | Sign of instructor with date | |  |  | | | | | | | | | | | | |

|  |
| --- |
| Practical  Course title: Big Data Analytics  Course term: 2021-2022 |
| Problem Statement: To implement K-means algorithm using map-reduce. |
| Theory: |
| 7.3. K-MEANS ALGORITHMS 255 7.3.1 K-Means Basics A k-means algorithm is outlined in Fig. 7.7. There are several ways to select the initial k points that represent the clusters, and we shall discuss them in Section 7.3.2. The heart of the algorithm is the for-loop, in which we consider each point other than the k selected points and assign it to the closest cluster, where “closest” means closest to the centroid of the cluster. Note that the centroid of a cluster can migrate as points are assigned to it. However, since only points near the cluster are likely to be assigned, the centroid tends not to move too much. Initially choose k points that are likely to be in different clusters; Make these points the centroids of their clusters; FOR each remaining point p DO find the centroid to which p is closest; Add p to the cluster of that centroid; Adjust the centroid of that cluster to account for p; END; Figure 7.7: Outline of k-means algorithms An optional step at the end is to fix the centroids of the clusters and to reassign each point, including the k initial points, to the k clusters. Usually, a point p will be assigned to the same cluster in which it was placed on the first pass. However, there are cases where the centroid of p’s original cluster moved quite far from p after p was placed there, and p is assigned to a different cluster on the second pass. In fact, even some of the original k points could wind up being reassigned. As these examples are unusual, we shall not dwell on the subject. 7.3.2 Initializing Clusters for K-Means We want to pick points that have a good chance of lying in different clusters. There are two approaches. 1. Pick points that are as far away from one another as possible. 2. Cluster a sample of the data, perhaps hierarchically, so there are k clusters. Pick a point from each cluster, perhaps that point closest to the centroid of the cluster. The second approach requires little elaboration. For the first approach, there are variations. One good choice is: Pick the first point at random; 256 CHAPTER 7. CLUSTERING WHILE there are fewer than k points DO Add the point whose minimum distance from the selected points is as large as possible; END; Example 7.8 : Let us consider the twelve points of Fig. 7.2, which we reproduce here as Fig. 7.8. In the worst case, our initial choice of a point is near the center, say (6,8). The furthest point from (6,8) is (12,3), so that point is chosen next. (4,10) (7,10) (9,3) (11,4) (12,6) (5,2)(2,2) (3,4) (4,8) (6,8) (10,5) (12,3) Figure 7.8: Repeat of Fig. 7.2 Among the remaining ten points, the one whose minimum distance to either (6,8) or (12,3) is a maximum is (2,2). That point has distance √52 = 7.21 from (6,8) and distance √101 = 10.05 to (12,3); thus its “score” is 7.21. You can check easily that any other point is less than distance 7.21 from at least one of (6,8) and (12,3). Our selection of three starting points is thus (6,8), (12,3), and (2,2). Notice that these three belong to different clusters. Had we started with a different point, say (10,5), we would get a different set of three initial points. In this case, the starting points would be (10,5), (2,2), and (4,10). Again, these points belong to the three different clusters. ✷ 7.3.3 Picking the Right Value of k We may not know the correct value of k to use in a k-means clustering. However, if we can measure the quality of the clustering for various values of k, we can usually guess what the right value of k is. Recall the discussion in Section 7.2.3, especially Example 7.5, where we observed that if we take a measure of appropriateness for clusters, such as average radius or diameter, that value will grow slowly, as long as the number of clusters we assume remains at or above the true number of clusters. However, as soon as we try to form fewer 7.3. K-MEANS ALGORITHMS 257 clusters than there really are, the measure will rise precipitously. The idea is expressed by the diagram of Fig. 7.9. Average Diameter Number of Clusters Correct value of k Figure 7.9: Average diameter or another measure of diffuseness rises quickly as soon as the number of clusters falls below the true number present in the data If we have no idea what the correct value of k is, we can find a good value in a number of clustering operations that grows only logarithmically with the true number. Begin by running the k-means algorithm for k = 1, 2, 4, 8, . . . . Eventually, you will find two values v and 2v between which there is very little decrease in the average diameter, or whatever measure of cluster cohesion you are using. We may conclude that the value of k that is justified by the data lies between v/2 and v. If you use a binary search (discussed below) in that range, you can find the best value for k in another log2 v clustering operations, for a total of 2 log2 v clusterings. Since the true value of k is at least v/2, we have used a number of clusterings that is logarithmic in k. Since the notion of “not much change” is imprecise, we cannot say exactly how much change is too much. However, the binary search can be conducted as follows, assuming the notion of “not much change” is made precise by some formula. We know that there is too much change between v/2 and v, or else we would not have gone on to run a clustering for 2v clusters. Suppose at some point we have narrowed the range of k to between x and y. Let z = (x + y)/2. Run a clustering with z as the target number of clusters. If there is not too much change between z and y, then the true value of k lies between x and z. So recursively narrow that range to find the correct value of k. On the other hand, if there is too much change between z and y, then use binary search in the range between z and y instead. 7.3.4 The Algorithm of Bradley, Fayyad, and Reina This algorithm, which we shall refer to as BFR after its authors, is a variant of k-means that is designed to cluster data in a high-dimensional Euclidean space. It makes a very strong assumption about the shape of clusters: they must be normally distributed about a centroid. The mean and standard deviation for a cluster may differ for different dimensions, but the dimensions must be |



|  |
| --- |
| Code: |
| Write map-reduce code to implement k-means algorithm code for mapper: |
| package exp6;  import java.io.IOException;  import org.apache.hadoop.io.\*;  import org.apache.hadoop.mapreduce.Mapper;  import exp6.Data;  public class Mapper6 extends Mapper<LongWritable, Text, Text, Text> {      public void map(LongWritable ikey, Text ivalue, Context context)  throws IOException, InterruptedException {    String value = ivalue.toString();  String st[] = value.split(",");    Double x = Double.parseDouble(st[0]);  Double y = Double.parseDouble(st[1]);    Double minDist=Double.MAX\_VALUE,dist=0.0;    String cluster = null;    if(Data.trackcluster < Data.maxCluster) {  Data.trackcluster++;  Data.Centers.add(new Coordinate(x,y,Integer.toString(Data.trackcluster)));  context.write(new Text(Integer.toString(Data.trackcluster)), new Text(Double.toString(x)+","+Double.toString(y)));    }else {  for(Coordinate co : Data.Centers) {  dist = Coordinate.euclidDist(co, x, y);  if(minDist > dist) {  minDist = dist;  cluster = co.name;  }  }  context.write(new Text(cluster),new Text(Double.toString(x)+","+Double.toString(y)));  }    }  } |
| Code for Reducer: |
| package exp6;  import java.io.IOException;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class Reducer6 extends Reducer<Text, Text, Text, Text> {  public void reduce(Text \_key, Iterable<Text> values, Context context) throws IOException, InterruptedException {  // process values  String[] st;    Double x\_avg=0.0,y\_avg=0.0;    StringBuilder sb = new StringBuilder();    int index = Integer.parseInt(\_key.toString());  double total = 0;    for(Text value:values) {  st = value.toString().split(",");    x\_avg+= Double.parseDouble(st[0]);  y\_avg+= Double.parseDouble(st[1]);    sb.append(value+" ");  total++;  }      Data.Centers.get(index-1).x = x\_avg/total;  Data.Centers.get(index-1).y = y\_avg/total;    context.write(new Text("C"+\_key.toString()+" ("+Data.Centers.get(index-1).x+","+Data.Centers.get(index-1).y+")"), new Text(sb.toString()));  }  } |
| Code for Driver Class: |
| package exp6;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class Driver6 {  public static void main(String[] args) throws Exception {    Configuration conf = new Configuration();  Path output = new Path(Data.root+"/output\_exp6");  FileSystem fs = FileSystem.get(output.toUri(),conf);  while(Data.iter < Data.nth\_iter) {    if(fs.exists(output)) {  fs.delete(output,true);  }    Job job = Job.getInstance(conf, "KMeans");  job.setJarByClass(Driver6.class);  // TODO: specify a mapper  job.setMapperClass(Mapper6.class);  // TODO: specify a reducer  job.setReducerClass(Reducer6.class);    job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(Text.class);    // TODO: specify output types  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(Text.class);  // TODO: specify input and output DIRECTORIES (not files)  FileInputFormat.setInputPaths(job, new Path(Data.root+"/input/kmean.txt"));  FileOutputFormat.setOutputPath(job, new Path(Data.root+"/output\_exp6"));  System.out.println(job.waitForCompletion(true)+" "+Data.iter++);  }  }  } |

|  |
| --- |
|  |

|  |
| --- |
| Explain BoW algorithm of clustering Answer for postlab question  A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.  The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.  A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:  A vocabulary of known words.  A measure of the presence of known words.  It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.  A very common feature extraction procedures for sentences and documents is the bag-of-words approach (BOW). In this approach, we look at the histogram of the words within the text, i.e. considering each word count as a feature.  The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.  The bag-of-words can be as simple or complex as you like. The complexity comes both in deciding how to design the vocabulary of known words (or tokens) and how to score the presence of known words. |