TU Berlin

Advanced Information Management

Homework assignment 1

Programming in Hadoop and Clustering Excercises

Author:
Ward Schodts

Supervisor:
Juan Soto

June 10, 2016

Programming in Hadoop

1. WordCount - "Hello World" of MapReduce

```java
package de.tuberlin.dima.aim3.assignment1;

import de.tuberlin.dima.aim3.HadoopJob;
import org.apache.hadoop.fs.Path;
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.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.function.Function;
import java.util.regex.Pattern;
import java.util.stream.StreamSupport;

import static java.util.stream.Collectors.counting;
import static java.util.stream.Collectors.groupingBy;

public class FilteringWordCount extends HadoopJob {

    @Override
    public int run(String[] args) throws Exception {
        Map<String, String> parsedArgs = parseArgs(args);

        Path inputPath = new Path(parsedArgs.get("--input"));
        Path outputPath = new Path(parsedArgs.get("--output"));

        Job wordCount = prepareJob(inputPath, outputPath, TextInputFormat.class,
    FilteringWordCountMapper.class,
                Text.class, IntWritable.class, WordCountReducer.class, Text.class,
    IntWritable.class, TextOutputFormat.class);

        wordCount.waitForCompletion(true);

        return 0;
    }

    static class FilteringWordCountMapper extends Mapper<Object, Text, Text,
    IntWritable> {

        private ArrayList<String> filterList = new ArrayList<>();

        /**
         * Method to add words that should be filtered out.
         *
         * @param fl : the list with filtered words
         */
        public void addWordsToFilter(List<String> fl) {
            this.filterList.addAll(fl);
        }

        public List getFilterList() {
            return this.filterList;
        }


```

```
60          @Override
61          protected void map(Object key, Text line, Context ctx) throws IOException,
    InterruptedException {
62              String[] filterList = {"to", "and", "in", "the"};
63              addWordsToFilter(Arrays.asList(filterList));
64              Pattern.compile(" ").splitAsStream(line.toString().replace(",", "")).map(
    String::toLowerCase).filter(l -> !getFilterList().contains(l)).collect(groupingBy(
    Function.identity(), counting())).forEach((word, count) -> writeToCtx(word, count,
    ctx));
65          }
66
67          private void writeToCtx(String word, Long val, Context ctx) {
68              try {
69                  ctx.write(new Text(word), new IntWritable(val.intValue()));
70              } catch (InterruptedException | IOException ignored) {
71
72              }
73          }
74
75      }
76
77      static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
     {
78
79          @Override
80          protected void reduce(Text key, Iterable<IntWritable> values, Context ctx)
81                  throws IOException, InterruptedException {
82
83              ctx.write(key, new IntWritable(StreamSupport.stream(values.spliterator(),
    false).mapToInt(IntWritable::get).sum()));
84          }
85
86      }
87
88  }
```

Listing 1: FilteringWordCount.java

```
1  16/06/08 23:46:45 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
       arguments. Applications should implement Tool for the same.
2  16/06/08 23:46:45 WARN mapred.JobClient: No job jar file set.  User classes may not be
       found. See JobConf(Class) or JobConf#setJar(String).
3  16/06/08 23:46:45 INFO input.FileInputFormat: Total input paths to process : 1
4  16/06/08 23:46:45 INFO mapred.JobClient: Running job: job_local_0001
5  16/06/08 23:46:45 INFO mapred.MapTask: io.sort.mb = 100
6  16/06/08 23:46:45 INFO mapred.MapTask: data buffer = 79691776/99614720
7  16/06/08 23:46:45 INFO mapred.MapTask: record buffer = 262144/327680
8  16/06/08 23:46:45 INFO mapred.MapTask: Starting flush of map output
9  16/06/08 23:46:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
       your platform... using builtin-java classes where applicable
10 16/06/08 23:46:45 INFO compress.CodecPool: Got brand-new compressor
11 16/06/08 23:46:45 INFO mapred.MapTask: Finished spill 0
12 16/06/08 23:46:45 INFO mapred.Task: Task:attempt_local_0001_m_000000_0 is done. And is
       in the process of commiting
13 16/06/08 23:46:46 INFO mapred.JobClient:  map 0% reduce 0%
14 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
15 16/06/08 23:46:48 INFO mapred.Task: Task 'attempt_local_0001_m_000000_0' done.
16 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
17 16/06/08 23:46:48 INFO mapred.Merger: Merging 1 sorted segments
18 16/06/08 23:46:48 INFO compress.CodecPool: Got brand-new decompressor
19 16/06/08 23:46:48 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left
        of total size: 80 bytes
20 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
21 16/06/08 23:46:48 INFO mapred.Task: Task:attempt_local_0001_r_000000_0 is done. And is
       in the process of commiting
22 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
```

```
16/06/08 23:46:48 INFO mapred.Task: Task attempt_local_0001_r_000000_0 is allowed to commit now
16/06/08 23:46:48 INFO output.FileOutputCommitter: Saved output of task 'attempt_local_0001_r_000000_0' to /tmp/mahout-FilteringWordCountTest-1822392791331859456/output
16/06/08 23:46:49 INFO mapred.JobClient:  map 100% reduce 0%
16/06/08 23:46:51 INFO mapred.LocalJobRunner: reduce > reduce
16/06/08 23:46:51 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' done.
16/06/08 23:46:52 INFO mapred.JobClient:  map 100% reduce 100%
16/06/08 23:46:52 INFO mapred.JobClient: Job complete: job_local_0001
16/06/08 23:46:52 INFO mapred.JobClient: Counters: 16
16/06/08 23:46:52 INFO mapred.JobClient:   Map-Reduce Framework
16/06/08 23:46:52 INFO mapred.JobClient:     Combine output records=0
16/06/08 23:46:52 INFO mapred.JobClient:     Spilled Records=34
16/06/08 23:46:52 INFO mapred.JobClient:     Map output materialized bytes=84
16/06/08 23:46:52 INFO mapred.JobClient:     Reduce input records=17
16/06/08 23:46:52 INFO mapred.JobClient:     Reduce output records=9
16/06/08 23:46:52 INFO mapred.JobClient:     Map input records=4
16/06/08 23:46:52 INFO mapred.JobClient:     SPLIT_RAW_BYTES=133
16/06/08 23:46:52 INFO mapred.JobClient:     Map output records=17
16/06/08 23:46:52 INFO mapred.JobClient:     Map output bytes=153
16/06/08 23:46:52 INFO mapred.JobClient:     Reduce shuffle bytes=0
16/06/08 23:46:52 INFO mapred.JobClient:     Combine input records=0
16/06/08 23:46:52 INFO mapred.JobClient:     Reduce input groups=9
16/06/08 23:46:52 INFO mapred.JobClient:   File Input Format Counters
16/06/08 23:46:52 INFO mapred.JobClient:     Bytes Read=107
16/06/08 23:46:52 INFO mapred.JobClient:   FileSystemCounters
16/06/08 23:46:52 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=64694
16/06/08 23:46:52 INFO mapred.JobClient:     FILE_BYTES_READ=672
16/06/08 23:46:52 INFO mapred.JobClient:   File Output Format Counters
16/06/08 23:46:52 INFO mapred.JobClient:     Bytes Written=78

Process finished with exit code 0
```

Listing 2: Output FilteringWordCount

2. A custom Writable

```java
package de.tuberlin.dima.aim3.assignment1;

import org.apache.hadoop.hdfs.util.ByteArray;
import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.Arrays;

public class PrimeNumbersWritable implements Writable {

    private int[] numbers;

    public PrimeNumbersWritable() {
        numbers = new int[0];
    }

    public PrimeNumbersWritable(int... numbers) {
        this.numbers = numbers;
    }

    @Override
    public void write(DataOutput out) throws IOException {
        out.writeInt(numbers.length);
        Arrays.stream(numbers).forEach((v) -> writeToOut(out, v));
    }

    private void writeToOut(DataOutput out, int i) {
```

```
        try {
            out.writeInt(i);
        } catch (IOException ignored) {
        }
    }


    @Override
    public void readFields(DataInput in) throws IOException {
        int length = in.readInt();

        int[] temp = new int[length];
        for (int i = 0; i < length; i++) {
            temp[i] = in.readInt();
        }
        this.numbers = temp;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof PrimeNumbersWritable) {
            PrimeNumbersWritable other = (PrimeNumbersWritable) obj;
            return Arrays.equals(numbers, other.numbers);
        }
        return false;
    }

    @Override
    public int hashCode() {
        return Arrays.hashCode(numbers);
    }
}
```

Listing 3: PrimeNumbersWritable.java

```
Process finished with exit code 0
```

Listing 4: Output PrimeNumbersWritable

3. Average temperature per month

```
package de.tuberlin.dima.aim3.assignment1;


import de.tuberlin.dima.aim3.HadoopJob;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.Map;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class AverageTemperaturePerMonth extends HadoopJob {

    @Override
    public int run(String[] args) throws Exception {
        Map<String, String> parsedArgs = parseArgs(args);

        Path inputPath = new Path(parsedArgs.get("--input"));
```

```
27          Path outputPath = new Path(parsedArgs.get("--output"));
28
29          double minimumQuality = Double.parseDouble(parsedArgs.get("--minimumQuality"));
30
31          Job averageTemperature = prepareJob(inputPath, outputPath, TextInputFormat.
        class,
32                  AverageTemperatureMapper.class, Text.class, IntWritable.class,
33                  AverageTemperatureReducer.class, Text.class, DoubleWritable.class,
34                  TextOutputFormat.class);
35
36          averageTemperature.getConfiguration().set("minimumQuality", String.valueOf(
        minimumQuality));
37
38          averageTemperature.waitForCompletion(true);
39
40          return 0;
41      }
42
43
44      static class AverageTemperatureMapper extends Mapper<Object, Text, Text,
        IntWritable> {
45
46
47          @Override
48          protected void map(Object key, Text line, Context ctx) throws IOException,
        InterruptedException {
49              Configuration conf = ctx.getConfiguration();
50              double minimumQuality = Double.parseDouble(conf.get("minimumQuality"));
51
52              String l = line.toString();
53              String[] ls = l.split("\t");
54              if (minimumQuality <= Double.parseDouble(ls[ls.length - 1])) {
55                  String K = ls[0] + "\t" + ls[1];
56                  ctx.write(new Text(K), new IntWritable(Integer.parseInt(ls[2])));
57              }
58
59          }
60
61      }
62
63      static class AverageTemperatureReducer extends Reducer<Text, IntWritable, Text,
        DoubleWritable> {
64          @Override
65          protected void reduce(Text key, Iterable<IntWritable> values, Context ctx)
66                  throws IOException, InterruptedException {
67              int sum = 0;
68              int length = 0;
69              for (IntWritable value : values) {
70                  sum += value.get();
71                  length++;
72              }
73              double average = (double) sum / length;
74              ctx.write(new Text(key), new DoubleWritable(average));
75          }
76      }
77  }
```

Listing 5: AverageTemperaturePerMonth.java

```
1  16/06/08 23:57:44 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
       arguments. Applications should implement Tool for the same.
2  16/06/08 23:57:44 WARN mapred.JobClient: No job jar file set.  User classes may not be
       found. See JobConf(Class) or JobConf#setJar(String).
3  16/06/08 23:57:44 INFO input.FileInputFormat: Total input paths to process : 1
4  16/06/08 23:57:44 INFO mapred.JobClient: Running job: job_local_0001
5  16/06/08 23:57:44 INFO mapred.MapTask: io.sort.mb = 100
```

```
16/06/08 23:57:44 INFO mapred.MapTask: data buffer = 79691776/99614720
16/06/08 23:57:44 INFO mapred.MapTask: record buffer = 262144/327680
16/06/08 23:57:44 INFO mapred.MapTask: Starting flush of map output
16/06/08 23:57:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
    your platform... using builtin-java classes where applicable
16/06/08 23:57:44 INFO compress.CodecPool: Got brand-new compressor
16/06/08 23:57:44 INFO mapred.MapTask: Finished spill 0
16/06/08 23:57:44 INFO mapred.Task: Task:attempt_local_0001_m_000000_0 is done. And is
    in the process of commiting
16/06/08 23:57:45 INFO mapred.JobClient:  map 0% reduce 0%
16/06/08 23:57:47 INFO mapred.LocalJobRunner:
16/06/08 23:57:47 INFO mapred.Task: Task 'attempt_local_0001_m_000000_0' done.
16/06/08 23:57:47 INFO mapred.LocalJobRunner:
16/06/08 23:57:47 INFO mapred.Merger: Merging 1 sorted segments
16/06/08 23:57:47 INFO compress.CodecPool: Got brand-new decompressor
16/06/08 23:57:47 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left
     of total size: 14587 bytes
16/06/08 23:57:47 INFO mapred.LocalJobRunner:
16/06/08 23:57:47 INFO mapred.Task: Task:attempt_local_0001_r_000000_0 is done. And is
    in the process of commiting
16/06/08 23:57:47 INFO mapred.LocalJobRunner:
16/06/08 23:57:47 INFO mapred.Task: Task attempt_local_0001_r_000000_0 is allowed to
    commit now
16/06/08 23:57:47 INFO output.FileOutputCommitter: Saved output of task '
    attempt_local_0001_r_000000_0' to /tmp/mahout-AverageTemperaturePerMonthTest
    -2969850880617501696/output
16/06/08 23:57:48 INFO mapred.JobClient:  map 100% reduce 0%
16/06/08 23:57:50 INFO mapred.LocalJobRunner: reduce > reduce
16/06/08 23:57:50 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' done.
16/06/08 23:57:51 INFO mapred.JobClient:  map 100% reduce 100%
16/06/08 23:57:51 INFO mapred.JobClient: Job complete: job_local_0001
16/06/08 23:57:51 INFO mapred.JobClient: Counters: 16
16/06/08 23:57:51 INFO mapred.JobClient:   Map-Reduce Framework
16/06/08 23:57:51 INFO mapred.JobClient:     Combine output records=0
16/06/08 23:57:51 INFO mapred.JobClient:     Spilled Records=14988
16/06/08 23:57:51 INFO mapred.JobClient:     Map output materialized bytes=14591
16/06/08 23:57:51 INFO mapred.JobClient:     Reduce input records=7494
16/06/08 23:57:51 INFO mapred.JobClient:     Reduce output records=72
16/06/08 23:57:51 INFO mapred.JobClient:     Map input records=10000
16/06/08 23:57:51 INFO mapred.JobClient:     SPLIT_RAW_BYTES=149
16/06/08 23:57:51 INFO mapred.JobClient:     Map output records=7494
16/06/08 23:57:51 INFO mapred.JobClient:     Map output bytes=84301
16/06/08 23:57:51 INFO mapred.JobClient:     Reduce shuffle bytes=0
16/06/08 23:57:51 INFO mapred.JobClient:     Combine input records=0
16/06/08 23:57:51 INFO mapred.JobClient:     Reduce input groups=72
16/06/08 23:57:51 INFO mapred.JobClient:   File Input Format Counters
16/06/08 23:57:51 INFO mapred.JobClient:     Bytes Read=192657
16/06/08 23:57:51 INFO mapred.JobClient:   FileSystemCounters
16/06/08 23:57:51 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=95989
16/06/08 23:57:51 INFO mapred.JobClient:     FILE_BYTES_READ=400317
16/06/08 23:57:51 INFO mapred.JobClient:   File Output Format Counters
16/06/08 23:57:51 INFO mapred.JobClient:     Bytes Written=1769

Process finished with exit code 0
```

Listing 6: Output AverageTemperature

Clustering

1. Metrics I

We have 3 points A(4,8), B(9,5) and C(2,2).
The *centroid* is then:

$$(x, y) = \left(\frac{4+9+2}{3}, \frac{8+5+2}{3}\right) = (5, 5)$$

We now calculate the error towards the centroid for every point:

A(4,8):

$$\sqrt{(4-5)^2 + (8-5)^2} = \sqrt{10}$$

B(9,5):

$$\sqrt{(9-5)^2 + (5-5)^2} = \sqrt{16} = 4$$

C(2,2):

$$\sqrt{(2-5)^2 + (2-5)^2} = \sqrt{18} = 3 \cdot \sqrt{2}$$

The SSE is then:

$$\sqrt{10}^2 + 4^2 + 3 \cdot \sqrt{2}^2 = 44$$

2. Metrics II

If you partition 3 points into 2 clusters, there's one with one element and one with two. The SSE of 1 point is always 0. So the total SSE is only determined by the SSE of the possible combinations of two points.

$$Centroid((3,0),(0,7)) = (1.5, 3.5)$$
$$SSE((3,0),(0,7)) = \sqrt{2.25 + 12.25}^2 + \sqrt{2.25 + 12.25}^2 = 29$$
$$Centroid(((0,7),(6,5)) = (3,6)$$
$$SSE(((0,7),(6,5)) = \sqrt{9+1}^2 + \sqrt{9+1}^2 = 20$$
$$Centroid((3,0),(6,5)) = (4.5, 2.5)$$
$$SSE((3,0),(6,5)) = \sqrt{2.25 + 6.25}^2 + \sqrt{2.25 + 6.25}^2 = 17$$

The SSE is the smallest in the last configuration therefore the the most optimal split in two groups is the following:

Cluster 1: [(0,7)]
Cluster 2: [(3,0),(6,5)]

3. CURE Algorithm

x = (0,0); y = (10,10), a = (1,6); b = (3,7); c = (4,3); d = (7,7), e = (8,2); f = (9,5) As was stated in the question, the two furthest points are x and y. So we start by calculating the distance to them.

$D(.,.)$	a	b	c	d	e	f
x	$\sqrt{37}$	$\sqrt{58}$	5	$\sqrt{98}$	$\sqrt{68}$	$\sqrt{106}$
y	$\sqrt{97}$	$\sqrt{58}$	$\sqrt{85}$	$\sqrt{18}$	$\sqrt{68}$	$\sqrt{26}$

Of all the points the minimum value for e is a maximum, so e is added first. $\rightarrow$ (b) is false.

Now we do the same again for e:

$D(.,.)$	a	b	c	d	e	f
x	$\sqrt{37}$	$\sqrt{58}$	5	$\sqrt{98}$	$\sqrt{68}$	$\sqrt{106}$
y	$\sqrt{97}$	$\sqrt{58}$	$\sqrt{85}$	$\sqrt{18}$	$\sqrt{68}$	$\sqrt{26}$
e	$\sqrt{65}$	$\sqrt{50}$	$\sqrt{17}$	$\sqrt{26}$	///	$\sqrt{10}$

Of all the points the minimum value for b is a maximum, so b is added next. $\rightarrow$ (d) is true.

We calculate the distance w.r.t. b:

$D(.,.)$	a	b	c	d	e	f
x	$\sqrt{37}$	$\sqrt{58}$	5	$\sqrt{98}$	$\sqrt{68}$	$\sqrt{106}$
y	$\sqrt{97}$	$\sqrt{58}$	$\sqrt{85}$	$\sqrt{18}$	$\sqrt{68}$	$\sqrt{26}$
e	$\sqrt{65}$	$\sqrt{50}$	$\sqrt{17}$	$\sqrt{26}$	///	$\sqrt{10}$
b	$\sqrt{5}$	///	$\sqrt{17}$	$\sqrt{16}$	$\sqrt{50}$	$\sqrt{40}$

Of all the points the minimum value for c is a maximum, so c is added third. $\rightarrow$ (a) and (c) are false.

We calculate the distance w.r.t. c:

$D(.,.)$	a	b	c	d	e	f
x	$\sqrt{37}$	$\sqrt{58}$	5	$\sqrt{98}$	$\sqrt{68}$	$\sqrt{106}$
y	$\sqrt{97}$	$\sqrt{58}$	$\sqrt{85}$	$\sqrt{18}$	$\sqrt{68}$	$\sqrt{26}$
e	$\sqrt{65}$	$\sqrt{50}$	$\sqrt{17}$	$\sqrt{26}$	///	$\sqrt{10}$
b	$\sqrt{5}$	///	$\sqrt{17}$	$\sqrt{16}$	$\sqrt{50}$	$\sqrt{40}$
c	$\sqrt{18}$	$\sqrt{17}$	///	5	$\sqrt{17}$	$\sqrt{29}$

Of all the points the minimum value for d is a maximum, so d is added fourth.

We calculate the distance w.r.t. d:

$D(.,.)$	a	b	c	d	e	f
x	$\sqrt{37}$	$\sqrt{58}$	5	$\sqrt{98}$	$\sqrt{68}$	$\sqrt{106}$
y	$\sqrt{97}$	$\sqrt{58}$	$\sqrt{85}$	$\sqrt{18}$	$\sqrt{68}$	$\sqrt{26}$
e	$\sqrt{65}$	$\sqrt{50}$	$\sqrt{17}$	$\sqrt{26}$	///	$\sqrt{10}$
b	$\sqrt{5}$	///	$\sqrt{17}$	4	$\sqrt{50}$	$\sqrt{40}$
c	$\sqrt{18}$	$\sqrt{17}$	///	5	$\sqrt{17}$	$\sqrt{29}$
d	$\sqrt{37}$	$\sqrt{16}$	$\sqrt{17}$	///	$\sqrt{26}$	$\sqrt{8}$

Of all the points,now the minimum value for f is a maximum, so f is added fifth. Obviously a is added as last.

4. A Comparative Analysis of Clustering Algorithms

	K-Means	CURE: Clustering Using Representatives	BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies	DBSCAN: Density-based Spatial Clustering of Applications with Noise
Type	• Partitioning method • Point assignment	• Partitioning • Hierarchical method • Random sampling	Hierarchical method	Density-based method
Space complexity	$\mathcal{O}((n+k)d)$	$\mathcal{O}(n)$	A little more than one scan of the data, i.e. almost linear.	$\mathcal{O}(n)$
Time complexity	$\mathcal{O}(kind)$ with i number of iterations, n number of points, k number of clusters and d the dimensionality of the data.	• $\mathcal{O}(n^2 \log n)$ • if the dimensionality is low it reduces to $\mathcal{O}(n^2)$	• $\mathcal{O}(n^3)$ • Priority queue: $\mathcal{O}(n^2 \log n)$	• $\mathcal{O}(n \log n)$: with KD-trees • $\mathcal{O}(n^2)$: without KD-trees
Applicability & advantages	• Easy and simple to implement. • You get pure subclusters if you specify a high enough number of clusters	• Clusters can have any shape • Assumes points are in euclidean space	• Very large data sets • Does not require whole data set in advance. • Can handle noise	• Can handle clusters of arbitrary shape • Scalability • Robust for outliers
Limitations & disadvantages	• Does not work for really big datasets. Some optimisation is needed then. • Cannot handle outliers • K needs to be predetermined	The algorithm cannot be directly applied to large databases because of the high runtime complexity.	• Cannot identify clusters that have non-spherical shapes • Cannot correct erroneous merges or splits	• Performance dependent on parameters • Not partitionable for multiprocessor systems
Assumptions	Clusters normally distributed	Points are in eucledian space	A changing order of iteration shouldn't have a too big influence on the output	No number of clusters needs to be assumed. Relies on density

References

[1] Hendrik Blockeel. "Machine Learning and Inductive Inference". In: *ACCO Leuven* (2010).

[2] *Clustering Algorithms: K-means*. http://www.cs.princeton.edu/courses/archive/spr08/cos435/Class_notes/clustering2_toPost.pdf. Accessed: June 2016.

[3] *CURE data clustering algorithm - Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/CURE_data_clustering_algorithm. Accessed: June 2016.

[4] *Data Clustering: A Review*. http://www.cs.rutgers.edu/~mlittman/courses/lightai03/jain99data.pdf. Accessed: June 2016.

[5] *DBSCAN - Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/DBSCAN. Accessed: June 2016.

[6] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[7] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "CURE: an efficient clustering algorithm for large databases". In: *ACM SIGMOD Record*. Vol. 27. 2. ACM. 1998, pp. 73–84.

[8] *k-means clustering Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/K-means_clustering. Accessed: June 2016.

[9] Juan Soto. *AIM3 – Scalable Data Analysis and DataMining*. https://isis.tuberlin.de/pluginfile.php/532517/mod_resource/content/1/05Clustering%202016.pdf. Accessed: June 2016.

[10] *Topic9: Density-based Clustering*. https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwiSxpyT-pvNAhUKWxQKHbcvBVYQFgguMAI&url=http%3A%2F%2Fwww2.cs.uh.edu%2F~ceick%2FML%2FTopic9.ppt&usg=AFQjCNFKy1_wYTyBBIJugNLQW08plGUgeA&sig2=14LQVkrcFJPbIgGVBiUnsA&bvm=bv.124088155,d.d24. Accessed: June 2016.