学院：数据科学与计算机学院　　专业：计算机科学与技术
姓名：郑 康 泽　　学号：17341213

云计算项目实践

课程设计——K-Means

一. 选题以及相应工作

我的选题是：基于Hadoop平台编写实现K-Means算法的MapReduce程序。主要工作是编写Java程序，特别是分清Map阶段和Reduce阶段应该实现什么功能。最后，通过自定义的测试集进行测试，初步确定实现的算法的正确性。

二. 原理分析

1. K-Means算法原理

以下是通过我的理解写出的算法描述：

Algorithm 1 K-Means

Input: Sample set $\mathbf{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$
Number of clusters k
Output: Cluster $\mathbf{C} = \{C_1, C_2, ..., C_n\}$
Procedure:

```
Randomly choose k samples from D as an initial mean vector group {μ_1, μ_2, ..., μ_n};
repeat
    Let C_i = ∅ (1 ≤ i ≤ k);
    for j = 1, 2, ..., m do
        Calculate the distance of sample x_j and each mean vector μ_i (1 ≤ i ≤ k): d_ji = ||x_j − μ_i||_2;
        Determine the cluster label of x_j according to the nearest mean vector: λ_j = arg min_{i∈{1,2,...,k}} d_ji;
        Add sample x_j into the corresponding cluster division C_λj = C_λj ∪ {x_j};
    end for
    for i = 1, 2, ..., k do
        Calculate new mean vector: μ'_i = 1/|C_i| Σ_{x∈C_i} x;
        if μ_i ≠ μ'_i then
            Update the current mean vector μ_i to μ'_i;
        else
            Keep the current mean vector unchanged;
        end if
    end for
until The current mean vectors have not changed
```

输入是一个数据集$\mathbf{D}$和一个数字k，数据集$\mathbf{D}$中由样本点$\mathbf{x}_j$构成的，或者说是样本的特征向量，k是要将数据集中的样本分成几类。首先我们从数据集中随机选出k个样本点分别作为这k类的中心（均值）$\boldsymbol{\mu}_i$，接下来执行一个循环。循环的第一个步是初始化k个集合为空集$\emptyset$，这些集合是来记录属于该类的样本的；循环的第二步是计算每一个样本点$\mathbf{x}_j$到每一个类的中心$\boldsymbol{\mu}_i$的距离d_{ji}，然后选择距离最小的类C_{λ_j}，将该样本点加入到距离最小的类C_{λ_j}中；循环的第三步是计算每个类的中心并更新。重复以上循环直到所有类的中心不再变化。

2. Map阶段的工作和Reduce阶段的工作
   Map阶段的工作是：计算每个样本点到每个类的中心距离，选择具有最小距离的类并加入，输出键值对为（**属于哪个类别的下标，样本点**）；
   Reduce阶段的工作是：重新计算每个类的中心，输出键值对为（**类中心，类下标**）。
3. 根据上面的描述，我们发现Map和Reduce只是完成一次循环，因此在main函数中要求能够判断是否停止循环以及重新执行循环，也即再次提交Job的功能。

三. 结果截图

1. 第一次测试如图
   第一个参数为作为输入的文件夹，第二个参数作为输出的文件夹，第三个参数为分类数k：

```
konzem@KONZEM:~/workspace2$ hadoop jar KMeans.jar KMeans ./input ./output 2
20/07/05 20:41:06 INFO Configuration.deprecation: session.id is deprecated. Ins
tead, use dfs.metrics.session-id
20/07/05 20:41:06 INFO jvm.JvmMetrics: Initializing JVM Metrics with processNam
e=JobTracker, sessionId=
20/07/05 20:41:06 INFO input.FileInputFormat: Total input paths to process : 1
20/07/05 20:41:06 INFO mapreduce.JobSubmitter: number of splits:1
20/07/05 20:41:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_l
ocal1234228904_0001
20/07/05 20:41:07 INFO mapred.LocalDistributedCacheManager: Creating symlink: /
tmp/hadoop-konzem/mapred/local/1593952866887/centroids.txt <- /home/konzem/work
space2/centroids.txt
20/07/05 20:41:07 INFO mapred.LocalDistributedCacheManager: Localized file:/hom
e/konzem/workspace2/centroids/centroids.txt as file:/tmp/hadoop-konzem/mapred/l
ocal/1593952866887/centroids.txt
20/07/05 20:41:07 INFO mapreduce.Job: The url to track the job: http://localhos
```

   运行完程序，当前文件夹下多出以下文件，其中centroids文件夹中记录了类中心，output-0和output-1文件夹为Reducer的输出，output-final为最后的分类结果。

```
konzem@KONZEM:~/workspace2$ ls
centroids       KMeans.java                 'KMeans$Utils.class'   test.txt
input           'KMeans$KMeansMapper.class'  output-0
KMeans.class    'KMeans$KMeansReducer.class' output-1
KMeans.jar      'KMeans$Parameters.class'    output-final
konzem@KONZEM:~/workspace2$
```

   测试集为6个5维的点，并且明显可以看出测试集可以分为2类；centroids文件夹中输出的两个类的中心也十分准确（输出格式为类中心 + 类下标）；output-final最后分类的结果也十分符合我们直观上的分类（输出格式为样本点 + 属于哪个类）。output-0对应的是Reducer第一次的输出，可以看出第一次循环后，就已经正确找到了每个类的中心（输出格式为类中心 + 类下标）；output-1对应的是Reducer第二次的输出，可

以看见类中心并没有发生变化，因此循环也就结束了，没有output-2出现。

```
konzem@KONZEM:~/workspace2$ cat ./input/test.txt
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
50 50 50 50 50
51 51 51 51 51
52 52 52 52 52
konzem@KONZEM:~/workspace2$ cat ./centroids/centroids.txt
2.0 2.0 2.0 2.0 2.0     0
51.0 51.0 51.0 51.0 51.0        1
konzem@KONZEM:~/workspace2$ cat ./output-final/final-data
1.0 1.0 1.0 1.0 1.0     0
2.0 2.0 2.0 2.0 2.0     0
3.0 3.0 3.0 3.0 3.0     0
50.0 50.0 50.0 50.0 50.0        1
51.0 51.0 51.0 51.0 51.0        1
52.0 52.0 52.0 52.0 52.0        1
konzem@KONZEM:~/workspace2$ cat ./output-0/part-r-00000
2.0 2.0 2.0 2.0 2.0     0
51.0 51.0 51.0 51.0 51.0        1
konzem@KONZEM:~/workspace2$ cat ./output-1/part-r-00000
2.0 2.0 2.0 2.0 2.0     0
51.0 51.0 51.0 51.0 51.0        1
```

2. 第二次测试如图：

```
konzem@KONZEM:~/workspace2$ hadoop jar KMeans.jar KMeans ./input ./output 3
20/07/05 21:04:07 INFO Configuration.deprecation: session.id is deprecated. Ins
tead, use dfs.metrics.session-id
20/07/05 21:04:07 INFO jvm.JvmMetrics: Initializing JVM Metrics with processNam
e=JobTracker, sessionId=
20/07/05 21:04:07 INFO input.FileInputFormat: Total input paths to process : 1
20/07/05 21:04:07 INFO mapreduce.JobSubmitter: number of splits:1
20/07/05 21:04:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_l
ocal253567524_0001
20/07/05 21:04:08 INFO mapred.LocalDistributedCacheManager: Creating symlink: /
tmp/hadoop-konzem/mapred/local/1593954247925/centroids.txt <- /home/konzem/work
space2/centroids.txt
20/07/05 21:04:08 INFO mapred.LocalDistributedCacheManager: Localized file:/hom
e/konzem/workspace2/centroids/centroids.txt as file:/tmp/hadoop-konzem/mapred/l
ocal/1593954247925/centroids.txt
20/07/05 21:04:08 INFO mapreduce.Job: The url to track the job: http://localhos
t:8080/
20/07/05 21:04:08 INFO mapreduce.Job: Running job: job_local253567524_0001
20/07/05 21:04:08 INFO mapred.LocalJobRunner: OutputCommitter set in config nul
l
20/07/05 21:04:08 INFO output.FileOutputCommitter: File Output Committer Algori
thm version is 1
20/07/05 21:04:08 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.had
oop.mapreduce.lib.output.FileOutputCommitter
20/07/05 21:04:08 INFO mapred.LocalJobRunner: Waiting for map tasks
20/07/05 21:04:08 INFO mapred.LocalJobRunner: Starting task: attempt_local25356
7524_0001_m_000000_0
```

```
konzem@KONZEM:~/workspace2$ ls
centroids     'KMeans$KMeansMapper.class'    output-1
input         'KMeans$KMeansReducer.class'   output-2
KMeans.class  'KMeans$Parameters.class'      output-final
KMeans.jar    'KMeans$Utils.class'           test.txt
KMeans.java    output-0
konzem@KONZEM:~/workspace2$
```

测试集为9个10维的点，并且明显可以测试集可以分为3类，结果也确实分为3类了。这次出现了output-2文件夹，说明循环了3次。

```
konzem@KONZEM:~/workspace2$ cat ./input/test.txt
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
50 50 50 50 50 50 50 50 50 50
51 51 51 51 51 51 51 51 51 51
52 52 52 52 52 52 52 52 52 52
100 100 100 100 100 100 100 100 100 100
101 101 101 101 101 101 101 101 101 101
102 102 102 102 102 102 102 102 102 102
konzem@KONZEM:~/workspace2$ cat ./centroids/centroids.txt
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 0
51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0       1
101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0     2
konzem@KONZEM:~/workspace2$ cat ./output-final/final-data
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 0
50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0       1
51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0       1
52.0 52.0 52.0 52.0 52.0 52.0 52.0 52.0 52.0 52.0       1
100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0     2
101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0     2
102.0 102.0 102.0 102.0 102.0 102.0 102.0 102.0 102.0 102.0     2
konzem@KONZEM:~/workspace2$ cat ./output-0/part-r-00000
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0
26.5 26.5 26.5 26.5 26.5 26.5 26.5 26.5 26.5 26.5       1
88.75 88.75 88.75 88.75 88.75 88.75 88.75 88.75 88.75 88.75     2
```

```
konzem@KONZEM:~/workspace2$ cat ./output-1/part-r-00000
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 0
51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0       1
101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0     2
konzem@KONZEM:~/workspace2$ cat ./output-2/part-r-00000
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 0
51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0 51.0       1
101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0 101.0     2
```

四. 遇到的问题以及解决的方法

1. 编程问题

   Java数组定义后，只能执行赋值操作，不能直接执行加等于号操作，否则会报未初始化的错误，如下程序会报错：

   ```
   Double[] test = new Double[10];
   for (int i=0; i<test.length; ++i)   test[i] += 1;
   ```

   必须先初始化，如下：

   ```
   Double[] test = new Double[10];
   for (int i=0; i<test.length; ++i)   test[i] = 0;
   for (int i=0; i<test.length; ++i)   test[i] += 1;
   ```

2. 设计问题

   Map阶段的需要用到的类中心从哪里来？为了解决这个问题，就必须设置一个文件存放类中心，然后在Map阶段时读取出来。此处就可以利用MapReduce的分布式缓存机制，利用该机制，可以让Mapper快速读取到

类中心。利用`job.addCacheFile`可以实现将文件放在缓存中，方便读取。

五. 仓库

仓库在此