

作业7

要求

在MapReduce上实现K-Means算法并在小数据集上测试。可以使用附件的数据集，也可以随机生成若干散点的二维数据 (x, y)。设置不同的K值和迭代次数，可视化聚类结果。

提交要求同作业5，附上可视化截图。

实现思路

我直接使用了实例代码来运行，用原来的代码创建maven项目KMeansExample。由于原来的代码不是用maven管理的，而且是基于Hadoop1.2编写的程序，所以有一些地方需要进行小小的修改。比如每个java文件前面都要加上对应的包名称，Job对象的创建需要调用getInstance静态方法，而不能直接new Job。

我尝试研读了整个算法的代码，下面简要描述一下示例代码的思路。

主程序：KMeansDriver.main()

KMeansDriver.main()方法是整个算法的主程序，它从命令行接收指定的参数k（需要聚成的类数），iterationNum（迭代次数），inputpath，outputpath。依次调用三个主要的过程：

generateInitialCluster(): 随机产生k个cluster center

clusterCenterJob(): 迭代更新cluster center

KMeansClusterJod(): 最终计算各个点所属的类

```
public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{
    System.out.println("start");
    Configuration conf = new Configuration();
    //命令行传入参数k,iterationNum,输入输出路径
    int k = Integer.parseInt(args[0]);
    int iterationNum = Integer.parseInt(args[1]);
    String sourcePath = args[2];
    String outputPath = args[3];
    KMeansDriver driver = new KMeansDriver(k, iterationNum, sourcePath,
    outputPath, conf);
    //随机生成k个cluster center
    driver.generateInitialCluster();
    System.out.println("initial cluster finished");
    //调用迭代的MapReduce过程，不断更新cluster center
    driver.clusterCenterJob();
    //迭代完成后，计算每个点所属的类
    driver.KMeansClusterJod();
}
```

更新簇中心：KMeans

在clusterCenterJob()方法中，通过循环提交KMeans job来实现cluster center的迭代更新。

```

public void clusterCenterJob() throws IOException, InterruptedException,
ClassNotFoundException{
    for(int i = 0;i < iterationNum; i++){
        Job clusterCenterJob = Job.getInstance(this.conf);
        ...
    }
}

```

KMeansMapper

KMeansMapper在setup方法中，读取上一次迭代/初始随机生成的cluster文件，生成k个Cluster类。

```

while((line = in.readLine()) != null){
    System.out.println("read a line:" + line);
    Cluster cluster = new Cluster(line);
    cluster.setNumOfPoints(0);
    kClusters.add(cluster);
}

```

在mapper方法中，对于每个样本点，计算样本点离哪个cluster center最近，然后构造包含该点的cluster，类别就是所属的类。

```

id = getNearest(instance);
if(id == -1)
    throw new InterruptedException("id == -1");
else{
    Cluster cluster = new Cluster(id, instance);
    cluster.setNumOfPoints(1);
    System.out.println("cluster that i emit is:" + cluster.toString());
    context.write(new IntWritable(id), cluster);
}

```

KMeansCombiner和KMeansReducer

KMeansCombiner对同类的cluster对象进行合并，将cluster center加权产生新的cluster center。

KMeansReducer和Combiner做的事情相同，也是将同类cluster对象合并，最后输出的就是更新后的cluster，和对应的cluster center坐标。

```

int numOfPoints = 0;
for(Cluster cluster : value){
    numOfPoints += cluster.getNumOfPoints();
    instance =
instance.add(cluster.getCenter().multiply(cluster.getNumOfPoints()));
}
Cluster cluster = new Cluster(key.get(),instance.divide(numOfPoints));

```

```
cluster.setNumOfPoints(numOfPoints);  
context.write(NullWritable.get(), cluster);
```

计算每个点所属的类: KMeansCluster

由于迭代过程生成的输出都是cluster center的信息, 所以最后判断每个点属于哪一个类还需要做一次MapReduce。KMeansCluster中, 只需要设计mapper方法, 对每一个点计算离它最近的cluster center, 对应的cluster id就是它的类别, reducer不需要做任何事, 最后直接将点的坐标和cluster id输出即可。

```
id = getNearest(instance);  
if(id == -1)  
    throw new InterruptedException("id == -1");  
else{  
    context.write(value, new IntWritable(id));  
}
```

运行和输出

运行需要指定的参数有: k, iterationNum, 输入路径, 输出路径。格式为:

hadoop jar [包名称] [KMeansDriver类名称] k iterationNum inputpath outputpath

我的作业中, 如果聚成3类, 迭代次数为3, 则命令为:

hadoop jar KMeansExample-1.0-SNAPSHOT.jar com.hw7.KMeansDriver 3 3 KMeans/input KMeans/output

最终聚类的结果在output\result1\clusteredInstances中。

运行截图

→ ↺ ⚠ 不安全 | node1:8088/cluster

100p

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
4	0	0	4	0	0 B	16 GB	0 B	0	16	0	2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1604651105148_0004	root	KMeansClusterJob	MAPREDUCE	default	Fri Nov 6 16:26:42 +0800 2020	Fri Nov 6 16:26:58 +0800 2020	FINISHED	SUCCEEDED	<div></div>	History	N/A
application_1604651105148_0003	root	clusterCenterJob2	MAPREDUCE	default	Fri Nov 6 16:26:22 +0800 2020	Fri Nov 6 16:26:40 +0800 2020	FINISHED	SUCCEEDED	<div></div>	History	N/A
application_1604651105148_0002	root	clusterCenterJob1	MAPREDUCE	default	Fri Nov 6 16:26:00 +0800 2020	Fri Nov 6 16:26:20 +0800 2020	FINISHED	SUCCEEDED	<div></div>	History	N/A
application_1604651105148_0001	root	clusterCenterJob0	MAPREDUCE	default	Fri Nov 6 16:25:40 +0800 2020	Fri Nov 6 16:25:58 +0800 2020	FINISHED	SUCCEEDED	<div></div>	History	N/A

全 | node1:19888/jobhistory/job/job_1604651105148_0004

Logged in as: dr.who

doop

MapReduce Job job_1604651105148_0004

Job Overview

Job Name: KMeansClusterJob

User Name: root

Queue: default

State: SUCCEEDED

Uberized: true

Submitted: Fri Nov 06 16:26:42 CST 2020

Started: Fri Nov 06 16:26:55 CST 2020

Finished: Fri Nov 06 16:26:58 CST 2020

Elapsed: 2sec

Diagnostics:

Average Map Time 1sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Nov 06 16:26:49 CST 2020	node1:8042	logs

Task Type	Total	Complete	
Map	1	1	
Reduce	0	0	
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	0

安全 | node1:19888/jobhistory/job/job_1604651105148_0003

🔍 ☆ 📄 ⚙️ 🌐



MapReduce Job job_1604651105148_0003

Job Overview			
Job Name: clusterCenterJob2			
User Name: root			
Queue: default			
State: SUCCEEDED			
Uberized: true			
Submitted: Fri Nov 06 16:26:22 CST 2020			
Started: Fri Nov 06 16:26:35 CST 2020			
Finished: Fri Nov 06 16:26:38 CST 2020			
Elapsed: 3sec			
Diagnostics:			
Average Map Time 0sec			
Average Shuffle Time 0sec			
Average Merge Time 0sec			
Average Reduce Time 1sec			

ApplicationMaster			
Attempt Number	Start Time	Node	Logs
1	Fri Nov 06 16:26:29 CST 2020	node2:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

```

finished!
20/11/06 16:26:42 INFO client.RMPProxy: Connecting to ResourceManager at node1/172.18.174.44:8032
20/11/06 16:26:42 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
r application with ToolRunner to remedy this.
20/11/06 16:26:42 INFO input.FileInputFormat: Total input paths to process : 1
20/11/06 16:26:42 INFO mapreduce.JobSubmitter: number of splits:1
20/11/06 16:26:42 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1604651105148_0004
20/11/06 16:26:42 INFO impl.YarnClientImpl: Submitted application application_1604651105148_0004
20/11/06 16:26:42 INFO mapreduce.Job: The url to track the job: http://node1:8088/proxy/application_1604651105148_0004/
20/11/06 16:26:42 INFO mapreduce.Job: Running job: job_1604651105148_0004
20/11/06 16:26:57 INFO mapreduce.Job: Job job_1604651105148_0004 running in uber mode : true
20/11/06 16:26:57 INFO mapreduce.Job: map 0% reduce 0%
20/11/06 16:26:59 INFO mapreduce.Job: map 100% reduce 0%
20/11/06 16:26:59 INFO mapreduce.Job: Job job_1604651105148_0004 completed successfully
20/11/06 16:26:59 INFO mapreduce.Job: Counters: 32
File System Counters

```

```

[root@node1 hw7exp1]# hdfs dfs -cat KMeans/output/clusteredInstances/*
86,43,3
5,36,2
16,58,2
66,47,3
20,37,2
89,27,3
56,68,1
21,42,2
96,22,3
72,80,1
99,10,3
20,74,1
59,19,3

```

设置不同的k和迭代次数并可视化

为了更加方便地设置不同k和iterationNum来运行。我直接写了一个脚本: generate_res.sh, 脚本自动迭代不同的参数, 然后将输出从hdfs中提取出来, 保存为csv文件。

```

#开始迭代, 从聚2个类开始
for((k=2;k<=$k_max;k++))

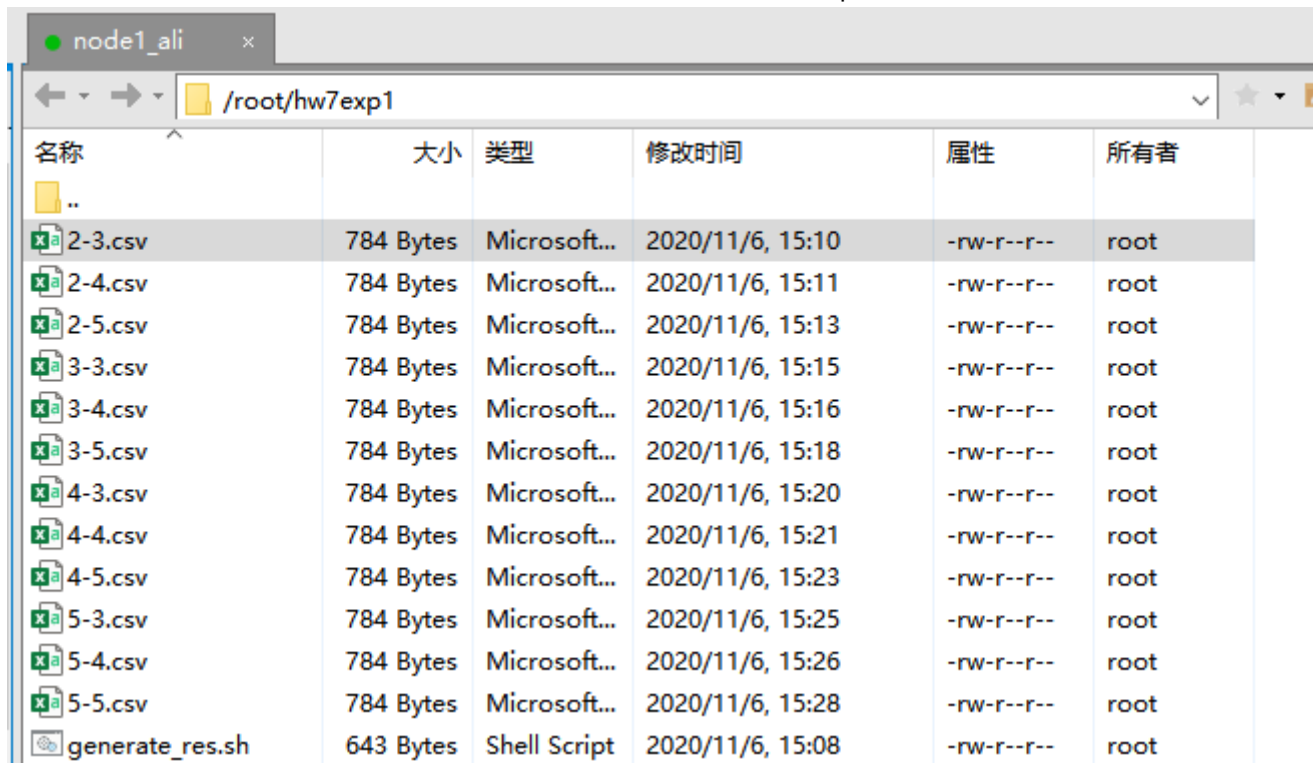
```

```

do
  for((n=3;n<=$iternum_max;n++))
  do
    hadoop jar KMeansExample-1.0-SNAPSHOT.jar com.hw7.KMeansDriver $k $n
    KMeans/input KMeans/output
    hdfs dfs -get KMeans/output/clusteredInstances/part-m-00000
    mv part-m-00000 $k-"$n".csv
    hdfs dfs -rm -r KMeans/output
  done
done

```

最后生成的都是文件名为"k-iterationNum"的csv文件(文件放在了output/result2中)



名称	大小	类型	修改时间	属性	所有者
2-3.csv	784 Bytes	Microsoft...	2020/11/6, 15:10	-rw-r--r--	root
2-4.csv	784 Bytes	Microsoft...	2020/11/6, 15:11	-rw-r--r--	root
2-5.csv	784 Bytes	Microsoft...	2020/11/6, 15:13	-rw-r--r--	root
3-3.csv	784 Bytes	Microsoft...	2020/11/6, 15:15	-rw-r--r--	root
3-4.csv	784 Bytes	Microsoft...	2020/11/6, 15:16	-rw-r--r--	root
3-5.csv	784 Bytes	Microsoft...	2020/11/6, 15:18	-rw-r--r--	root
4-3.csv	784 Bytes	Microsoft...	2020/11/6, 15:20	-rw-r--r--	root
4-4.csv	784 Bytes	Microsoft...	2020/11/6, 15:21	-rw-r--r--	root
4-5.csv	784 Bytes	Microsoft...	2020/11/6, 15:23	-rw-r--r--	root
5-3.csv	784 Bytes	Microsoft...	2020/11/6, 15:25	-rw-r--r--	root
5-4.csv	784 Bytes	Microsoft...	2020/11/6, 15:26	-rw-r--r--	root
5-5.csv	784 Bytes	Microsoft...	2020/11/6, 15:28	-rw-r--r--	root
generate_res.sh	643 Bytes	Shell Script	2020/11/6, 15:08	-rw-r--r--	root

每个csv文件中，第一列和第二列是样本的两个维度，第三列是对应的类，用matlab画出分类的效果图即可。下面是一些例子：

