

实验 3：MapReduce 和 Spark 编程实验

171830046 陆天淳

基本需求：利用用户购物数据集，分析预测回头客。涉及数据预处理、存储、查询、排序、统计分析等数据处理全流程所涉及的各种典型应用场景，涵盖 Hadoop、MapReduce、Spark、Hive、HBase 等系统和软件的安装和使用方法。

阶段一：MapReduce 编程

主要思路：将书中给的 WordCount.java 文件进行相应修改，完成对应要求。

完整代码见 WordCount.java

a. 数据读取

```
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {
    String[] userinfo=value.toString().split(",");
    Text word =new Text();
    word.set(uerinfo[10]);
    String product=uerinfo[1];
    Integer a = Integer.parseInt(uerinfo);
    int m=a.intValue();
    IntWritable i = new IntWritable();
```

b. 数据统计

```
public static <K, V extends Comparable<? super V>> Map<K, V> sortDescend(Map<K, V> map) {
    List<Map.Entry<K, V>> list = new ArrayList<>(map.entrySet());
    Collections.sort(list, new Comparator<Map.Entry<K, V>>() {
        public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) {
            int compare = (o1.getValue()).compareTo(o2.getValue());
            return -compare;
        }
    });
    Map<K, V> result = new HashMap<>();
    for (Map.Entry<K, V> entry : list) {
        result.put(entry.getKey(), entry.getValue());
    }
    return result;
}
```

1. 统计各省的双十一前十热门关注产品（“点击+添加购物车+购买+关注”总量最多前 10 的产品）

完整数据见 guanzhu.txt

| | | | | | |
|-----|-----------|-----------|-----------|------------|--------|
| 上海市 | 67897上海市 | 783997上海市 | 636863上海市 | 1024557上海市 | 628774 |
| 上海市 | 696384上海市 | 191499上海市 | 768758上海市 | 846404上海市 | 458784 |
| 云南 | 67897云南 | 783997云南 | 636863云南 | 1024557云南 | 770668 |
| 云南 | 628774云南 | 458784云南 | 846404云南 | 768758云南 | 94609 |
| 内蒙古 | 67897内蒙古 | 783997内蒙古 | 636863内蒙古 | 1024557内蒙古 | 770668 |
| 内蒙古 | 628774内蒙古 | 458784内蒙古 | 846404内蒙古 | 217788内蒙古 | 191499 |
| 北京市 | 67897北京市 | 783997北京市 | 636863北京市 | 1024557北京市 | 770668 |

2. 统计各省的双十一前十热门销售产品（购买最多前 10 的产品）

完整数据见 goumai.txt

| | | | | | |
|-----|-----------|-----------|------------|------------|--------|
| 上海市 | 191499上海市 | 353560上海市 | 514725上海市 | 1059899上海市 | 713695 |
| 上海市 | 735931上海市 | 67897上海市 | 1039919上海市 | 944554上海市 | 107407 |
| 云南 | 191499云南 | 1059899云南 | 349999云南 | 1010145云南 | 655904 |
| 云南 | 48664云南 | 147751云南 | 496758云南 | 81360云南 | 181387 |

阶段二：Hive 操作

1. 把精简数据集导入到数据仓库 Hive 中，并对数据仓库 Hive 中的数据进行查询分析

```
hive> create table users
> (user_id int,
> item_id int,
> cat_id int,
> merchant_id int,
> brand_id int,
> month int,
> day int,
> action int,
> age_range int,
> gender int,
> province string
> )
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde' WITH SERDEPROPERTIES
> (
> "separatorChar"=", "
> )
> STORED AS TEXTFILE;
OK
Time taken: 0.312 seconds
```

2. 查询双 11 那天有多少人购买了商品

```
hive> select count(1) from users where action =2
> ;
Query ID = root_20191220071009_c8679cf1-829e-42ef-ad4f-6c8248e2c0d6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1576687152693_0001, Tracking URL = http://h01:8088/proxy/
application_1576687152693_0001/
Kill Command = /usr/local/hadoop/bin/mapred job -kill job_1576687152693_000
1
Hadoop job information for Stage-1: number of mappers: 1; number of reducers
: 1
2019-12-20 07:10:53,759 Stage-1 map = 0%, reduce = 0%
2019-12-20 07:11:54,441 Stage-1 map = 0%, reduce = 0%, Cumulative CPU 12.53
sec
2019-12-20 07:12:09,889 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 19.
13 sec
2019-12-20 07:12:28,001 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2
2.99 sec
MapReduce Total cumulative CPU time: 22 seconds 990 msec
Ended Job = job_1576687152693_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 22.99 sec HDFS Read: 47
321376 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 22 seconds 990 msec
OK
116856
```

3. 查询双 11 那天男女买家购买商品的比例

```
hive> select count(1) from users where action =2
> ;
Query ID = root_20191220071009_c8679cf1-829e-42ef-ad4f-6c8248e2c0d6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1576687152693_0001, Tracking URL = http://h01:8088/proxy/
application_1576687152693_0001/
Kill Command = /usr/local/hadoop/bin/mapred job -kill job_1576687152693_000
1
Hadoop job information for Stage-1: number of mappers: 1; number of reducers
: 1
2019-12-20 07:10:53,759 Stage-1 map = 0%, reduce = 0%
2019-12-20 07:11:54,441 Stage-1 map = 0%, reduce = 0%, Cumulative CPU 12.53
sec
```

女性用户:

```
Total MapReduce CPU Time Spent: 5 seconds 410 msec
OK
39058
```

男性用户:

```
Total MapReduce CPU Time Spent: 5 seconds 430 msec
OK
38932
```

男女比为 0.997

4. 查询双 11 那天浏览次数前十的品牌

```
MapReduce Total cumulative CPU time: 3 seconds 320 msec
Ended Job = job_1576687152693_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 13.25 sec HDFS Read: 47
321274 HDFS Write: 128928 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 3.32 sec HDFS Read: 136
Amazon Write: 307 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 570 msec
OK
1360 49151
3738 10130
82 9719
1446 9426
6215 8568
1214 8470
5376 8282
2276 7990
1662 7808
8235 7661
Time taken: 89.221 seconds, Fetched: 10 row(s)
```

阶段三：Spark 编程

在助教的镜像基础上配置了 pyspark, 用 python 进行编程。

```
Spark context web UI available at http://10.1.40.40
Spark context available as 'sc' (master = local[*], app id = local-15769456
02).
Spark session available as 'spark'.
Welcome to

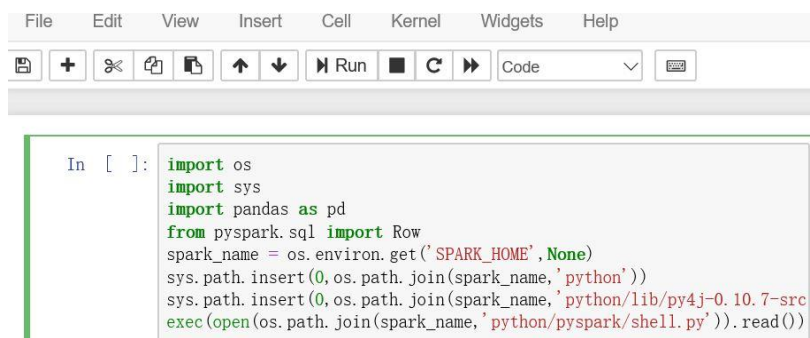
  ____  __
 / ___/  / /
/ /   /  / /
/ /___/  / /
/_____/  / /
         / /
        /_/

version 2.4.4

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
```

由于在 linux 中编程界面十分不友好, 这里使用了 JUPYTER 进行编程。相关配置参考 <https://www.jianshu.com/p/91365f343585>

配置完成后界面如下:



```
File Edit View Insert Cell Kernel Widgets Help
[Icons] [Run] [Code]
In [ ]: import os
import sys
import pandas as pd
from pyspark.sql import Row
spark_name = os.environ.get('SPARK_HOME', None)
sys.path.insert(0, os.path.join(spark_name, 'python'))
sys.path.insert(0, os.path.join(spark_name, 'python/lib/py4j-0.10.7-src
exec(open(os.path.join(spark_name, 'python/pyspark/shell.py')).read())
```

首先将数据导入 dataframe。Dataframe 极大的方便了后续的数据处理。

```
user=sc.textFile("file:///usr/local/homework/million_user_log.txt")

users=user.map(lambda x: x.split(','))

users=user.map(lambda x: Row(user_id=x[0], item_id=x[1], cat_id=x[2], merchant_id=x[3], brand_id=x[4], month=x[5],
                             day=x[6], action=x[7], age_range=x[8], gender=x[9], province=x[10]))

record=spark.createDataFrame(users)
```

1. 统计各省销售最好的产品类别前十（销售最多前 10 的产品类别）

```
action2 = record.select(record['action'] == '2')

province = record.select('province').distinct().collect()

for i in province:

    temp=action2.filter(record['province']==i[0])

    temp0=temp.groupby('cat_id').count()

    temp=temp0.sort('count', ascending=False).limit(10).collect()

    print(i[0])    //打印省份名称

    print(temp)
```

北京市[Row(cat_id='656', count=131), Row(cat_id='602', count=111), Row(cat_id='1208', count=101), Row(cat_id='177', count=9)
广东[Row(cat_id='656', count=132), Row(cat_id='1208', count=118), Row(cat_id='602', count=95), Row(cat_id='1401', count=87),
云南[Row(cat_id='1208', count=118), Row(cat_id='656', count=107), Row(cat_id='177', count=95), Row(cat_id='1142', count=82),
内蒙古[Row(cat_id='1208', count=118), Row(cat_id='656', count=105), Row(cat_id='662', count=92), Row(cat_id='602', count=86)
湖北[Row(cat_id='656', count=121), Row(cat_id='1208', count=112), Row(cat_id='1213', count=89), Row(cat_id='602', count=87),

完整数据见 spark1.txt

2. 统计各省的双十一热门销售产品（购买最多前 10 的产品）-- 和 MapReduce 作业对比结果

```
for j in province:

    temp=action2.filter(record['province']==j[0])

    temp0=temp.groupby('item_id').count()

    temp=temp0.sort('count', ascending=False).limit(10).collect()

    print(j[0])

    print(temp)
```

北京市[Row(item_id='1059899', count=8), Row(item_id='191499', count=8), Row(item_id='514725', count=6), Row(item_id='944554', count=6),
广东[Row(item_id='181387', count=7), Row(item_id='926069', count=7), Row(item_id='110347', count=6), Row(item_id='1059899', count=6),
云南[Row(item_id='191499', count=10), Row(item_id='1059899', count=7), Row(item_id='655904', count=5), Row(item_id='48664', count=5),
内蒙古[Row(item_id='353560', count=8), Row(item_id='191499', count=8), Row(item_id='770668', count=6), Row(item_id='1039999', count=6),
新疆[Row(item_id='81360', count=7), Row(item_id='191499', count=6), Row(item_id='107407', count=6), Row(item_id='147751', count=6),
海南[Row(item_id='1059899', count=10), Row(item_id='349999', count=7), Row(item_id='353560', count=7), Row(item_id='110347', count=7),
西藏[Row(item_id='191499', count=11), Row(item_id='353560', count=8), Row(item_id='1059899', count=7), Row(item_id='315360', count=7),
陕西[Row(item_id='191499', count=9), Row(item_id='353560', count=9), Row(item_id='514725', count=8), Row(item_id='936203', count=8),

完整数据见 spark2.txt

与 mapreduce 对比：

| | | | | | |
|-----|-----------|-----------|------------|------------|--------|
| 上海市 | 191499上海市 | 353560上海市 | 514725上海市 | 1059899上海市 | 713695 |
| 上海市 | 735931上海市 | 67897上海市 | 1039919上海市 | 944554上海市 | 107407 |
| 云南 | 191499云南 | 1059899云南 | 349999云南 | 1010145云南 | 655904 |
| 云南 | 48664云南 | 147751云南 | 496758云南 | 81360云南 | 181387 |

排序有些许差异，结果一致。

3. 查询双 11 那天浏览次数前十的品牌 -- 和 Hive 作业对比结果

```
action0 = record.filter(record['action'] == '0')

result0 = action0.groupby('brand_id').count()

result = result0.sort('count', ascending=False).limit(10).collect()
|
print(result)

[Row(brand_id='1360', count=49151), Row(brand_id='3738',
count=10130), Row(brand_id='82', count=9719), Row(brand_
id='1446', count=9426), Row(brand_id='6215', count=856
8), Row(brand_id='1214', count=8470), Row(brand_id='537
6', count=8282), Row(brand_id='2276', count=7990), Row(b
```

与 hive 对比：

```
1360 49151
3738 10130
82 9719
1446 9426
6215 8568
1214 8470
5376 8282
2276 7990
1662 7808
8235 7661
Time taken: 89.221 seconds, Fetched: 10 row(s)
```

结果一致。

阶段四：数据挖掘

针对预处理后的训练集和测试集，基于 MapReduce 或 Spark MLlib 编写程序预测回头客。

基本思路：采用支持向量机 SVM 分类器预测回头客。原本准备使用 pyspark 中的 ML 库进行预测，但是由于对这个库一无所知，出现了太多 bug……，转为使用 scala 进行编写。在程序的最后阶段数据保存阶段，多次 saveAsTextFile 失败以后转为打印所有数据，由于 ubuntu 命令行界面只能显示一部分数据，故手动复制黏贴了后面的数据到 out.txt 中，下面只展示了前 10 个数据。

1. Spark 启动

```
root@h01: /usr/local/spark-2.4.4/bin# ./spark-shell --jars /usr/local/spark-2.4.4/jars/mysql-connector-java-5.1.40-bin.jar
2019-12-24 18:19:31.909 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://h01:4040
Spark context available as 'sc' (master = local[*], app id = local-1577211590600).
Spark session available as 'spark'.
Welcome to

      ____
     / ___/
    / __/
   /___/
  version 2.4.4

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
Type in expressions to have them evaluated.
Type :help for more information.
```

2. 在 spark-shell 中导入 SVM 需要的包

```
scala> import org.apache.spark.SparkConf
import org.apache.spark.SparkConf

scala> import org.apache.spark.SparkContext
import org.apache.spark.SparkContext

scala> import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.regression.LabeledPoint

scala> import org.apache.spark.mllib.linalg.{Vectors, Vector}
import org.apache.spark.mllib.linalg.{Vectors, Vector}

scala> import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}

scala> import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics

scala> import java.util.Properties
import java.util.Properties

scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._

scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row
```

3. 读取数据

```
scala> val train_data = sc.textFile("file:///usr/local/train_after.csv")
train_data: org.apache.spark.rdd.RDD[String] = file:///usr/local/train_after.csv
MapPartitionsRDD[1] at textFile at <console>:35

scala> val test_data = sc.textFile("file:///usr/local/test_after.csv")
test_data: org.apache.spark.rdd.RDD[String] = file:///usr/local/test_after.csv
MapPartitionsRDD[3] at textFile at <console>:35

scala> val train = train_data.map(line =>
  | val parts = line.split(',')
  | LabeledPoint(parts(4).toDouble, Vectors.dense(parts(1).toDouble, parts(2)
  | .toDouble, parts(3).toDouble))
  | )
train: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[4] at map at <console>:36

scala> val test = test_data.map(line =>
  | val parts = line.split(',')
  | LabeledPoint(parts(4).toDouble, Vectors.dense(parts(1).toDouble, parts(2)
  | .toDouble, parts(3).toDouble))
  | )
test: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[5] at map at <console>:36
```

4. 构造 SVM 模型

```
scala> val model = SVMWithSGD.train(train, numIterations)
[Stage 0:===== (1 + 0) /

2019-12-24 18:23:46,610 WARN classification.SVMWithSGD: The input data is not
System Settings hed, which may hurt performance if its parent RDDs are also uncach
ed.
2019-12-24 18:23:48,533 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeSystemBLAS
2019-12-24 18:23:48,534 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeRefBLAS
2019-12-24 18:24:18,182 WARN classification.SVMWithSGD: The input data was not
directly cached, which may hurt performance if its parent RDDs are also uncac
hed.
model: org.apache.spark.mllib.classification.SVMModel = org.apache.spark.mllib
.classification.SVMModel: intercept = 0.0, numFeatures = 3, numClasses = 2, th
reshold = 0.0
```

5. 设置迭代次数并训练

```
scala> val numIterations = 1000
numIterations: Int = 1000

scala> val model = SVMWithSGD.train(train, numIterations)
[Stage 0:===== (1 + 0) /

2019-12-24 18:23:46,610 WARN classification.SVMWithSGD: The input data is not
directly cached, which may hurt performance if its parent RDDs are also uncach
ed.
2019-12-24 18:23:48,533 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeSystemBLAS
2019-12-24 18:23:48,534 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeRefBLAS
2019-12-24 18:24:18,182 WARN classification.SVMWithSGD: The input data was not
directly cached, which may hurt performance if its parent RDDs are also uncac
hed.
model: org.apache.spark.mllib.classification.SVMModel = org.apache.spark.mllib
.classification.SVMModel: intercept = 0.0, numFeatures = 3, numClasses = 2, th
reshold = 0.0
```

6. 模型评估

```
scala> model.clearThreshold()
res0: model.type = org.apache.spark.mllib.classification.SVMModel: intercept =
0.0, numFeatures = 3, numClasses = 2, threshold = None
```

7. 数据保存与展示

```
scala> scoreAndLabels.repartition(1).saveAsTextFile("file:///usr/local/out.txt
")
[Stage 1003:> (0 + 1) /

scala> val out=sc.textFile("file:///usr/local/out.txt")
out: org.apache.spark.rdd.RDD[String] = file:///usr/local/out.txt MapPartition
sRDD[2016] at textFile at <console>:35

scala> out.show()
<console>:37: error: value show is not a member of org.apache.spark.rdd.RDD[St
ring]
    out.show()
      ^

scala> out.take(10).foreach(println)
-31210.67145173342 1.0
-66350.78396289758 1.0
-34387.83425597505 1.0
-11804.239618771628 1.0
-38751.56686603498 1.0
-51998.201725312414 1.0
-67317.70791130241 1.0
-59539.089024148525 1.0
-6651.257745453658 1.0
-33795.58530548574 1.0
```