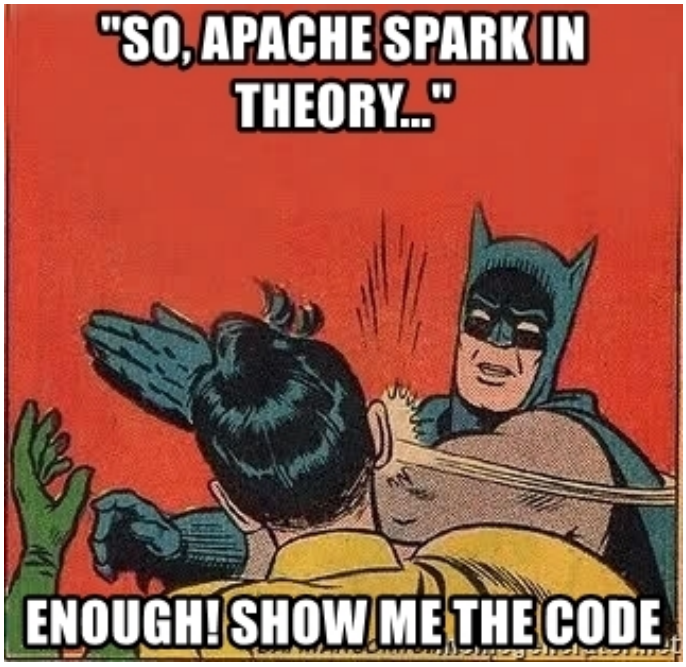
实验三 Spark机器学习



一、实验目标

本次实验旨在教会学生使用Spark 处理数据并实现机器学习算法。具体如下：

- 学习使用Spark-shell 基本命令（1 分）；
- 使用Spark 实现词频统计、计算数据方差（4 分）；
- 使用Spark 实现线性回归训练算法（6 分）。

在开始实验之前，请确认自己理解了RDD 概念，即掌握了理论课的内容。如果想了解更多，请参考RDD Programming Guide和Spark。

签到规则：完成第二-五题给助教检查 或者 到达下课时间 即可签到

二、学习Spark-shell常用指令

基本功能介绍

Spark-shell 是一个强大的交互式分析数据工具，同时也是一种学习Spark 的有效工具，它可以使用Scala 或Python 编写，本课程主要介绍Scala。开启spark-shell：

```
2020214210@thumm01:~$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
Spark Context Web UI is available at Spark Master Public URL
Spark context available as 'sc' (master = spark://thumm01:7077, app id = app-
20211017105714-0002).
Spark session available as 'spark'.
Welcome to
```

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.4
      /_/

Using Scala version 2.12.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_221)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Spark-shell 除了可以使用Scala 语言操作外，还有一些基本指令，这些指令都以 `:` 开头，指令的用法可以使用`:help` 查看：

```
scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:completions <string>    output completions for the given string
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show)
:h? <string>             search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the interpreter
:replay [options]        reset the repl and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the repl to its initial state, forgetting all session
entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:settings <options>      update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <type>        display the kind of a type. see also :help kind
:warnings                show the suppressed warnings from the most recent line which
had any
```

其中，常用指令为：

- `:quit` 退出spark-shell 控制台；
- `:load <path>` 加载使用scala 编写的spark-shell 脚本；
- `:save <path>` 将当前上下文的历史指令保存为文件。

当我们在spark-shell 中调试程序完成后，我们可以使用`:save`指令将历史保存到一个文件，我们可以使用这个文件恢复之前的调试的上下文，也能将其做成一个具有一定功能的脚本。

使用`:load`打印Hello Word

在ubuntu自己的目录下新建一个Scala文件，写入输出`Hello World`的语句，进入`spark-shell`使用`:load`运行该文件。（本题 1分）

三、使用Spark 进行词频统计

本题请同学们使用自己的数据集，这里需要先将数据集传入Hadoop文件系统里。请将输出结果放入实验报告。（本题 2 分）

首先，进入spark-shell。

```
2020214210@thumm01 :~$ spark-shell
```

接下来，我们需要加载待统计词频的数据集，输入以下内容：

```
scala> val words = sc.textFile("/dsjxtjc/2020214210/wc_dataset.txt")
words: org.apache.spark.rdd.RDD[String] = /dsjxtjc/2020214210/wc_dataset.txt
MapPartitionsRDD[1] at textFile at <console>:24
```

这句命令中，`sc`（Spark-Context）是spark-shell 的上下文，这个变量是进入spark-shell 就有的，可以用来设置一些运行参数；`val words`是定义一个变量名为words 的变量，它的值是使用`sc.textFile`函数加载HDFS 中words.txt 文件的内容。接下来查看words 的内容：

```
scala > words.first() # 查看第一行
res0: String = chapter
scala > words.count() # 查看行数
res1: Long = 2683500
```

使用一行代码统计词频：

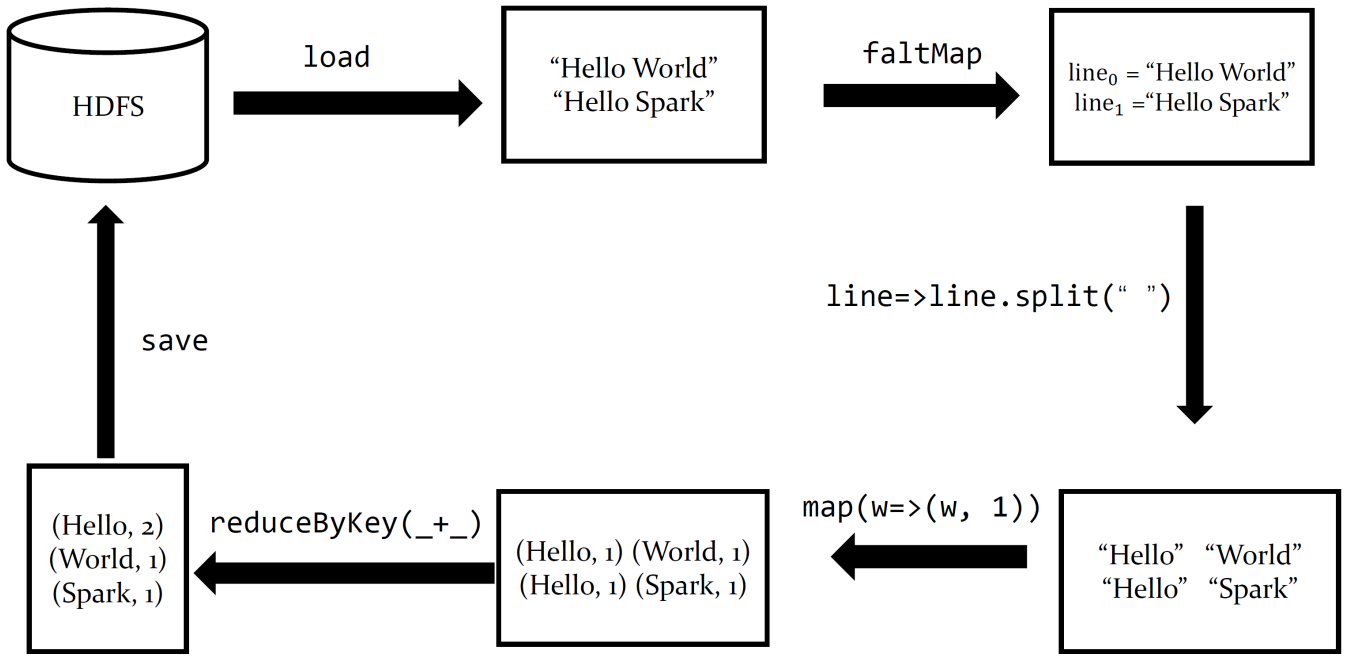
```
scala > val result = words.flatMap(l => l.split(" ")).map(w => (w, 1)).reduceByKey(_ +
_)
result: org.apache.spark.rdd.RDD[(String , Int)] = ShuffledRDD [4] at reduceByKey at
<console>:25
scala > result.first() # 查看结果的第一行内容
res3: (String, Int) = (someone, 100)
scala > result.saveAsTextFile("/dsjxtjc/2020214210/wc_output")
```

接下来我们来解释这行代码，代码的流程如下图所示，可分为以下步骤：

- 从HDFS 中加载文件`words.txt`，保存内容到变量`words`；
- 对`words`逐行处理，对每一行按空格进行分割，得到一个字符串列表；
- 使用`map`将字符串列表转成一个键值对列表`[<key1, value1>, <key2, value2>, ……]`，其中键为单词，

值为词频（没有合并之前为1）；

- 将不同的键值对根据相同的键不断地合并，直至无法合并，得到词频统计结果；
- 将结果保存到HDFS 中(保存到了`/dsjxtjc/学号/wc_output`)。



Bonus

换用另一种RDD 函数组合实现WordCount 功能（每种组合0.5 分，1 分封顶），测试不同实现方式的所需时间。

- 时间：`var t = new Date().getTime`

四、使用Spark 计算均值与方差

实验一要求大家实现自己的MapReduce 框架，然后使用自己的框架来计算均值与方差，接下来我们使用Spark 来实现这个功能。请用自己的数据集完成本题，并将运行结果截图放入报告。（本题 2 分）

首先，创建一个数据集(1 到1000,000)：

```
2020214210@thumm01:~$ for ((i=1; i<=1000000; i=i+1)); do echo $i >> numbers.txt; done
2020214210@thumm01:~$ tail -n 3 numbers.txt
999998
999999
1000000
```

这个数据集的均值为500000.5, 方差为288675.1345946685。接着将这个数据集上传到HDFS：

```
2020214210@thumm01 :~$ hadoop fs -put numbers.txt /dsjxtjc/2020214210/numbers.txt
2020214210@thumm01 :~$ hadoop fs -tail /dsjxtjc/2020214210/numbers.txt
… …
999999
1000000
```

运行spark-shell, 从HDFS 加载`number.txt`：

```
scala > val numbers = sc.textFile("/dsjxtjc/2020214210/numbers.txt")
numbers: org.apache.spark.rdd.RDD[String] = numbers.txt MapPartitionsRDD [1] at
textFile at <console >:24
```

加载的数据为字符串形式，需要转成数值型，这里转`double`型：

```
scala > val numbers_double = numbers.map(num => num.toDouble)
numbers_double: org.apache.spark.rdd.RDD[Double] = MapPartitionsRDD [2] at map at
<console >:25
```

统计数字个数：

```
scala > val n_num = numbers_double.count()
n_num: Long = 1000000
```

计算均值：

```
scala > val mean = numbers_double.reduce(_ + _) / n_num
mean: Double = 500000.5
```

计算方差：

```
scala > val variance = numbers_double.map(num => num - mean).map(num =>
num*num).reduce(_ + _) / n_num
variance: Double = 8.333333333359143E10
```

计算标准差：

```
scala > import scala.math._ # 导入scala 数学库
import scala.math._
scala > val std = sqrt(variance)
std: Double = 288675.1345952599
```

五、Spark 机器学习

现在大家已经学会了Spark 的基本操作，接下来请大家使用Scala实现线性回归，请使用自己的数据集完成本题。禁止使用任何MLlib 的库函数。（本题 6 分）

1. 一元线性回归给 3 分；
2. $n\,(n>1)$ 元线性回归（非向量运算，即使用循环逐个更新参数）给 4 分；
3. 向量表示的 $n\,(n>1)$ 元线性回归给 6 分。

提示：

- 应包括矩阵加法、矩阵减法、矩阵乘法、矩阵转置等接口；
- 需要截图放出每一轮的loss，如果训练轮次太多，不一定要全部截图，只需要证明它在下降即可；

签到点

六、报告提交要求

请严格按照以下要求提交实验报告。

1. 简要描述数据集含义，用**粗体**标出有多少条数据，多少GB 和数据集存放路径；
2. 将命令和结果截图（或复制输入输出）放入报告，实验报告需为pdf 格式（命名为`学号_姓名_实验三.pdf`，例如：`2021200000_张三_实验三.pdf`），连同代码文件一同打包成压缩文件（命名为`学号_姓名_实验三.*`，例如：`2021200000_张三_实验三.zip`），最后上交到网络学堂。压缩文件中文件目录应为（示例）：

```
.
├── code
│   ├── MultivariateLR.scala
│   ├── convert_enery_to_ijk.py
│   ├── energy.xlsx
│   ├── wordcout_bonus.scala
│   ├── xtest.txt
│   ├── xtrain.txt
│   ├── ytest.txt
│   └── ytrain.txt
└── 2020214210_解书照_实验三.pdf

1 directory, 9 files
```

3. 本次实验所有题目都不会用到root权限。
4. 迟交作业一周以内，以50% 比例计分；一周以上不再计分。另一经发现抄袭情况，零分处理。