

# 实验四：淘宝回头客预测

Design by W.H Huang | Direct by Prof Feng

## 1 实验目的

通过本次实验，你应该：

- 熟悉基于 spark 分布式编程环境
- 了解 svm 算法并掌握 mllib 相关 svm 相关函数使用
- 独立完成基于 svm 算法淘宝回头客预测

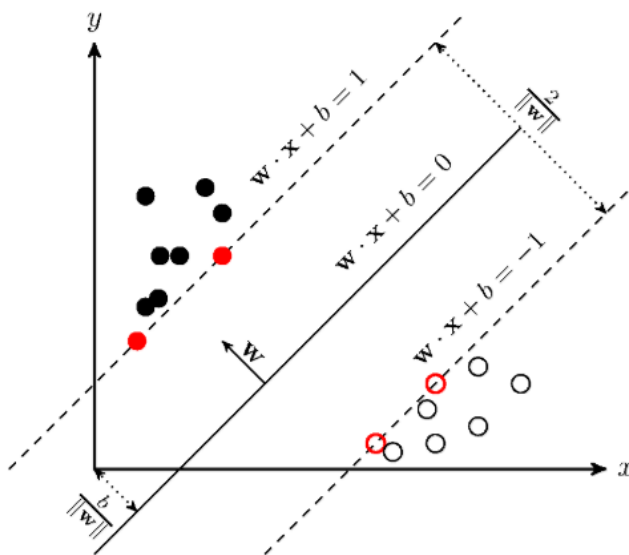
本次实验，你将根据代码相关提示完成整个 svm 模型建立、模型评估过程。

### 1.1 SVM 简介

支持向量机，因其英文名为 *support vector machine*，故一般简称 svm。svm 从线性可分情况下的最优分类面发展而来。

**最优分类面** 就是要求分类线不但能将两类正确分开(训练错误率为 0)，且使分类间隔最大。

svm 考虑寻找一个满足分类要求的超平面，并且使训练集中的点距离分类面尽可能的远，也就是寻找一个分类面使它两侧的空白区域(*margin*)最大。这两类样本中离分类面最近，且平行于最优分类面的超平面上的点，就叫做支持向量(下图中红色的点)。



假设超平面可描述为：

$$wx + b = 0, w \in R^n, b \in R$$

其分类间隔等于  $\frac{2}{\|w\|}$ 。其学习策略是使数据间的间隔最大化，最终转化为一个凸二次规划问题求解。

分类器的损失函数 (hinge loss 铰链损失) 如下所示：

$$L(w; x, y) := \max(0, 1 - yw^T x)$$

默认情况下，线性 SVM 是用 L2 正则化来训练的，但也支持 L1 正则化。在这种情况下，这个问题就变成了一个线性规划。

线性 SVM 算法输出一个 SVM 模型。给定一个新的数据点，比如说  $x$ ，这个模型就会根据  $W^T x$  的值来进行预测。默认情况下，如果  $W^T x \geq 0$ ，则输出预测结果为正（因为我们想要损失函数最小，如果预测为负，则会导致损失函数大于1），反之则预测为负。

## 2 实验准备

本次实验数据集在 `/home/hadoop/Experiment/Ex4_CustomerForecast/src` 下：

- `test.csv`：测试集

	A	B	C	D	E	F	G	H	I
1	user_id	age_range	gender	merchant_id	label				
2	34176	6	0	944	-1				
3	34176	6	0	412	-1				
4	34176	6	0	1945	-1				
5	34176	6	0	4752	-1				

相关字段及意义如下：

- `user_id`：买家id
- `age_range`：买家年龄分段
  - 1表示年龄小于 18
  - 2表示年龄在 [18, 24]
  - 3表示年龄在 [25, 29]
  - 4表示年龄在 [30, 34]
  - 5表示年龄在 [35, 39]
  - 6表示年龄在 [40, 49]
  - 7和8表示年龄大于等于 50
  - 0和NULL 则表示未知
- `gender`：性别，0 表示女性，1 表示男性，2和NULL 表示未知
- `merchant_id`：商家id
- `label`：是否是回头客，0 值表示不是回头客，1 值表示回头客，-1 值不考虑
- `train.csv`：训练集，字段同 `train.csv`，不再赘述。

	A	B	C	D	E	F	G	H
1	user_id	age_range	gender	merchant_id	label			
2	163968	0	0	4378	-1			
3	163968	0	0	2300	-1			
4	163968	0	0	1551	-1			
5	163968	0	0	4343	-1			

### 2.1 数据预处理

#### 2.1.1 训练集预处理

剔除掉 `train.csv` 中 字段值部分字段值为空的数据。

## 1. 创建 .sh 脚本

```
cd /home/hadoop/Experiment/Ex4_CustomerForecast/  
sed -i '1d' ./src/train.csv      # 删除训练集第一行  
touch predeal_train.sh          # 创建相应训练集处理脚本  
chmod +x ./predeal_train.sh     # 增加权限可写  
vim ./predeal_train.sh          # 开始编辑脚本
```

在打开的脚本中复制下列代码：

```
#!/bin/bash  
#下面设置输入文件，把用户执行predeal_train.sh命令时提供的第一个参数作为输入文件名称  
infile=$1  
#下面设置输出文件，把用户执行predeal_train.sh命令时提供的第二个参数作为输出文件名称  
outfile=$2  
#注意！！最后的$infile > $outfile必须跟在}'这两个字符的后面  
awk -F "," 'BEGIN{  
    id=0;  
}  
{  
    if($1 && $2 && $3 && $4 && ($5!=-1)){  
        id=id+1;  
        print $1,"$2","$3","$4","$5  
        if(id==1000000){ # 提取训练集前100w条label不等于-1（无效值）记录  
            exit  
        }  
    }  
}' $infile > $outfile
```

## 2. 开始处理

```
./predeal_train.sh ./src/train.csv ./src/train_after.csv
```

### 2.1.2 测试集预处理

只保留 test.csv 数据集里 label==1 字段，其余值剔除掉。使得测试集需要预测的 label 全为 1。

#### 1. 创建 .sh 脚本

```
cd /home/hadoop/Experiment/Ex4_CustomerForecast/  
touch predeal_test.sh          # 创建相应测试集处理脚本  
chmod +x ./predeal_test.sh     # 增加权限可写  
vim predeal_test.sh            # 开始编辑脚本
```

在打开的脚本中复制下列代码：

```
#!/bin/bash  
#下面设置输入文件，把用户执行predeal_test.sh命令时提供的第一个参数作为输入文件名称  
infile=$1  
#下面设置输出文件，把用户执行predeal_test.sh命令时提供的第二个参数作为输出文件名称  
outfile=$2
```

#注意!! 最后的\$infile > \$outfile必须跟在}'这两个字符的后面

```
awk -F "," 'BEGIN{
    id=0;
}
{
    if($1 && $2 && $3 && $4 && !$5){
        id=id+1;
        print $1,$2,$3,$4,"1"
        if(id==10000){
            exit
        }
    }
}' $infile > $outfile
```

## 2. 开始处理

```
./predeal_test.sh ./src/test.csv ./src/test_after.csv
```

## 2.2 上传 hdfs

😊 hdfs 常用相关操作可参考: [hdfs常用操作](#)

hdfs 是一个分布式文件系统, 现在你将需要在正式实验前将相关数据集上传 hdfs 文件系统上。

### 1. 创建文件夹

```
cd /usr/local/hadoop
bin/hadoop fs -mkdir -p /ex4/dataset
```

### 2. 上传数据集

将本地文件 train.csv 、 test.csv 上传到 hdfs://master:9000/ex4/dataset/ 下。

📁 以下上传的 hdfs 路径可简写为: /ex4/dataset

```
bin/hadoop fs -put /home/hadoop/Experiment/Ex4_CustomerForecast/src/test_after.csv
/ex4/dataset
bin/hadoop fs -put /home/hadoop/Experiment/Ex4_CustomerForecast/src/train_after.csv
/ex4/dataset
```

## 3 完成编码

在实验开始之前, 我们强烈建议你按照以下流程完成实验:

1. 命令行 下完成代码 单元测试
2. 单元测试无误, 将代码填充在相应给出的 py 文件函数中
3. spark-submit 方式提交代码

😊 如何在命令行下完成单元测试?

### 1. 启动 pyspark

△ 本次实验都是在集群环境下, 集群启动 pyspark 应该按以下方式:

- 先启动 Hadoop/Spark 集群

```
# 启动hadoop集群
cd /usr/local/hadoop
sbin/start-all.sh
# 启动spark集群
cd /usr/local/spark
sbin/start-master.sh
sbin/start-slaves.sh
```

- 启动 pyspark

```
bin/pyspark --master spark://master:7077
```

## 2. 命令行下单元测试

通过命令行下实时交互式体验，来快速测试代码是否正确。

具体实例可参考 ex2 中示例。

## 3. 提交代码

在命令行下单元测试后，便可以填写在相应 py 文件中。

本次实验通过 spark-submit 方式提交代码至 集群，请参照实验 3.2.1 部分进行。

# 3.1 forecast.py

forecast.py 可在服务器路径 /Ex4\_CustomerForecast/forecast.py 下编辑：



```
"""
@author: huangwanghui
@time: 2020/2/1 11:50
"""

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.classification import SVMWithSGD

conf = SparkConf().setAppName("ex4").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN") # 设置日志级别
spark = SparkSession(sc)

TRAINDATAPATH = "/ex4/dataset/train_after.csv"
TESTDATAPATH = "/ex4/dataset/test_after.csv"
```

相关函数及功能如下：

- GetParts：将读取的数据集 转换为标准 table-feature 形式。如：
  - ['1','2','3','4','0'] --> LabeledPoint(0, [2.0,3.0,4.0])
- Getpoint：预测并返回结果(元组)
- predict：SVMWithSGD 方式 预测淘宝回头客整体流程

现在请根据提示，完成 predict 函数，使得整体流程得于完成：

```

"""
@author: huangwanghui
@time: 2020/2/1 11:50
"""

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.classification import SVMWithSGD

# 命令行下以下不用设置, 已存在相应实例
conf = SparkConf().setAppName("ex4").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN") # 设置日志级别
spark = SparkSession(sc)

TRAINDATAPATH = "/ex4/dataset/train_after.csv"
TESTDATAPATH = "/ex4/dataset/test_after.csv"

def GetParts(line):
    """
    将读取的数据集train_after.csv、test_after.csv 转换为标准lable-feature形式, 例如:
    ['1','2','3','4','0'] --> LabeledPoint(0, [2.0,3.0,4.0])
    :param line: 读取的字符串
    :return:
    """
    parts = line.split(',')
    return
    LabeledPoint(float(parts[4]), Vectors.dense(float(parts[1]), float(parts[2]), float(parts[3])))
)

def Getpoint(model, point):
    """
    预测并返回结果(元组)
    :param model: 训练集构建的SVMWithSGD模型
    :param point: 测试集数据, 标准LabeledPoint类型数据
    :return: (测试集预测得分, 原始标签)
    """
    score = model.predict(point.features)
    return (score, point.label)

def predict(Iterations=1000, threshold=-90000):
    """
    SVMWithSGD 预测淘宝回头客整体流程
    :return:
    """

    # 【完成下面代码缺失处】

    # 1. 读取数据
    train_data = sc.textFile( )
    test_data = sc.textFile( )

```

```

# 2.数据标准化
train = train_data.map(lambda line: )
test = test_data.map(lambda line: )

# 3.构建模型
model = # 默认迭代1000次

# 4.评估模型
model.setThreshold(threshold) # 默认阈值 -90000
scoreAndLabels = test.map(lambda point: )
# 计算精度
accuracy = scoreAndLabels.filter(lambda l: ).count() / test.count()

return accuracy

if __name__ == "__main__" :

    accuracy = predict()
    print("accuracy: "+ str(predict()*100) + "%")

```

## 3.2 集群运行

### 3.2.1 集群运行任务

按照以下步骤启动集群运行任务：

#### 1. 启动集群

⚠ 启动集群下 pyspark 已启动集群则略过这步。

启动 hadoop 集群

```
cd /usr/local/hadoop
sbin/start-all.sh
```

启动 spark 集群

```
cd /usr/local/spark
sbin/start-master.sh
sbin/start-slaves.sh
```

#### 2. 上传集群运行任务

提交代码：

```
cd /usr/local/spark
bin/spark-submit --master spark://master:7077 --executor-memory 1G
/home/hadoop/Experiment/Ex4_CustomerForecast/forecast.py
```

#### 3. 运行过程

一切正常，你将会看到运行过程中打印出最终预测精度为 71.44%：

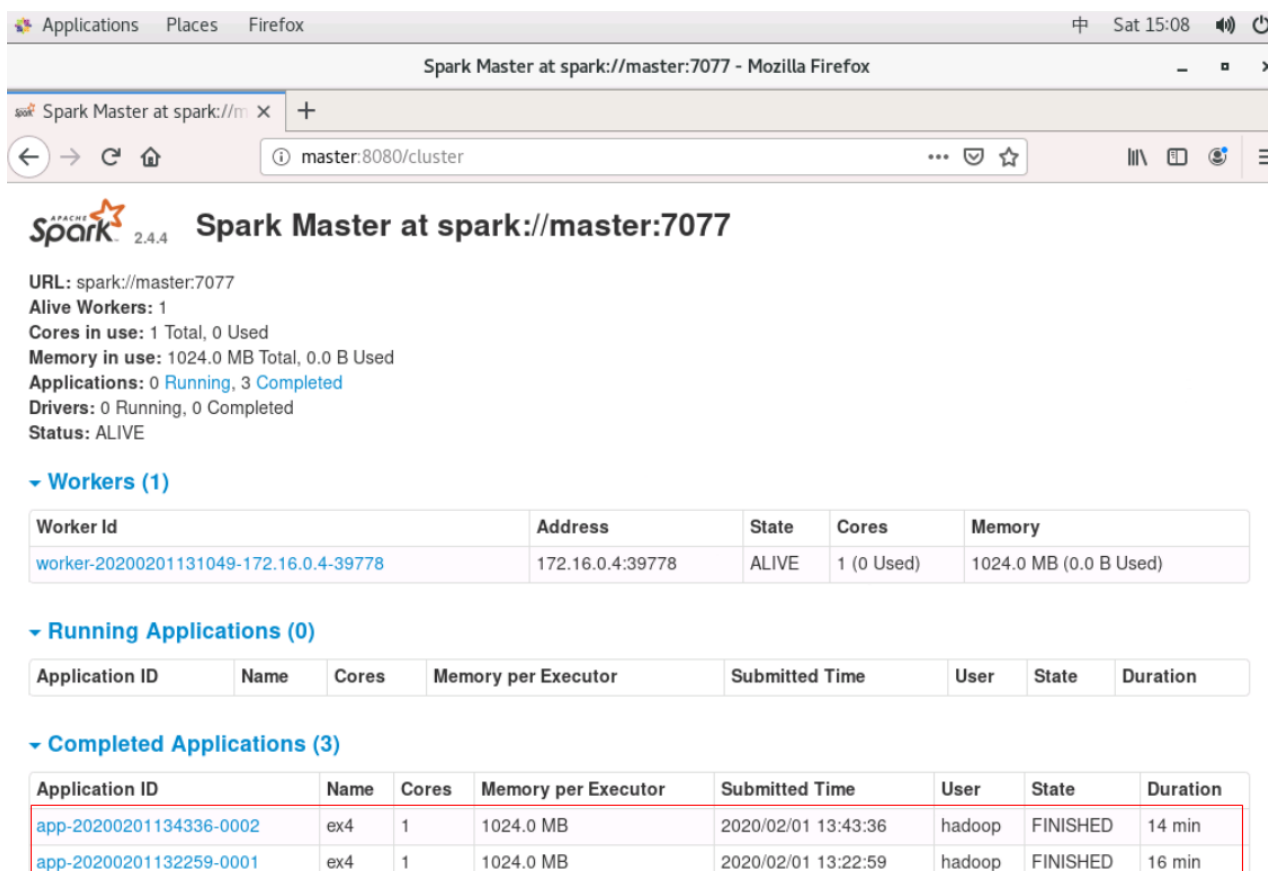
```

20/02/01 13:43:36 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, master,
33333, None)
20/02/01 13:43:37 INFO StandaloneSchedulerBackend: SchedulerBackend is ready for scheduling be
ginning after reached minRegisteredResourcesRatio: 0.0
20/02/01 13:43:55 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check yo
ur cluster UI to ensure that workers are registered and have sufficient resources
accuracy: 71.44%

```

#### 4. web UI 查看

Master 服务器输入: `master:8080`, 可查看此前运行的应用进程信息:



Spark Master at spark://master:7077

URL: spark://master:7077  
 Alive Workers: 1  
 Cores in use: 1 Total, 0 Used  
 Memory in use: 1024.0 MB Total, 0.0 B Used  
 Applications: 0 Running, 3 Completed  
 Drivers: 0 Running, 0 Completed  
 Status: ALIVE

▼ Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20200201131049-172.16.0.4-39778	172.16.0.4:39778	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

▼ Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20200201134336-0002	ex4	1	1024.0 MB	2020/02/01 13:43:36	hadoop	FINISHED	14 min
app-20200201132259-0001	ex4	1	1024.0 MB	2020/02/01 13:22:59	hadoop	FINISHED	16 min

### 3.2.2 结果分析

根据运行结果我们知道最终精度为 71.44%。经过实际多次测试, 可通过以下方式提高预测精度:

- 增大训练集: 修改 `predeal_train.sh` 脚本, 截取更多数据
- 增大迭代次数: `Iterations` 最大次数可由 1000 修改为 10000, 但是实际运行时间明显变长, 经过测试大约需要 40min。

如果你尝试打印 `scoreAndLabels`, 将得到类似以下结果:

第一列数字为 `SVM` 预测的测试集特征得分, 第二列数字为测试集对应特征标签

```

.....
-59045.132228013084 1.0
-81550.17634254562 1.0
-87393.69932070676 1.0
.....

```



### 3.3 常见集群错误

**ERROR:** slave01 节点上运行 jps 命令出现 information unavailable

```
[hadoop@slave01 local]$ jps
6035 -- process information unavailable
23703 -- process information unavailable
23482 -- process information unavailable
```

解决流程（由于是在 slave01 节点出现错误，以下都是在 slave01 进行操作！）：

1. 检查 /tmp 目录

```
cd /tmp
```

切换到 /tmp 目录下，我们发现存在 hsperfdata\_hadoop、hsperfdata\_root 相关文件：

```
[hadoop@slave01 local]$ cd /tmp/
[hadoop@slave01 tmp]$ ll
total 56
-rw-r--r-- 1 root root 6431 Jan 23 16:12 cvm init.log
drwxrwxrwx 2 hadoop hadoop 4096 Feb 1 10:06 hsperfdata_hadoop
drwxr-xr-x 2 root root 4096 Jan 26 20:16 hsperfdata_root
drwxrwxrwx 3 hadoop hadoop 4096 Jan 31 18:12 Jetty localhost_36244_datanode_851tna
drwxrwxr-x 3 hadoop hadoop 4096 Feb 1 09:53 Jetty localhost_39722_datanode____.85wgso
```

2. 删除相关文件

我们删除上述 hsperfdata\_\* 文件：

```
rm -rf hsperfdata_*
```

3. 重启集群

在 master 节点上再次重新 关闭 --> 启动 集群。

在 slave01 节点上运行 jps 命令，发现相关问题已被解决，进程信息可以被正常显示

```
[hadoop@slave01 local]$ jps
31204 Worker
9748 Jps
30011 DataNode
28363 NodeManager
```

## 4 实验小结

通过这次实验，你再一次体验了在 spark 分布式编程环境进行代码编写、测试，也独立完成了 svm 进行简单分类、预测。这也是我们的最后一次实验，希望你能通过前面学习有所收获~

由于一些主观、客观原因，我们实验条件比较一般、时间有限，实验设计并未从完全从头开始，而是让大家在理解的基础上对已有代码进行填充、完善。麻雀虽小，五脏俱全~相信你应该掌握了基本 spark 分布式编程技巧，当然集群出现的各种错误一定也让你抓耳挠腮印象深刻。

也许刚从新手村出来的你也还有很多疑问，我也没法一一解答，所以屠龙的勇士继续前进吧，毕竟搬砖的路还很长，与你共勉。