# 机器学习06【机器学习】

# 主要内容

1. 理解推荐系统处理数据流程。
2. python文件预处理Hive数据。
3. dubbo服务使用。

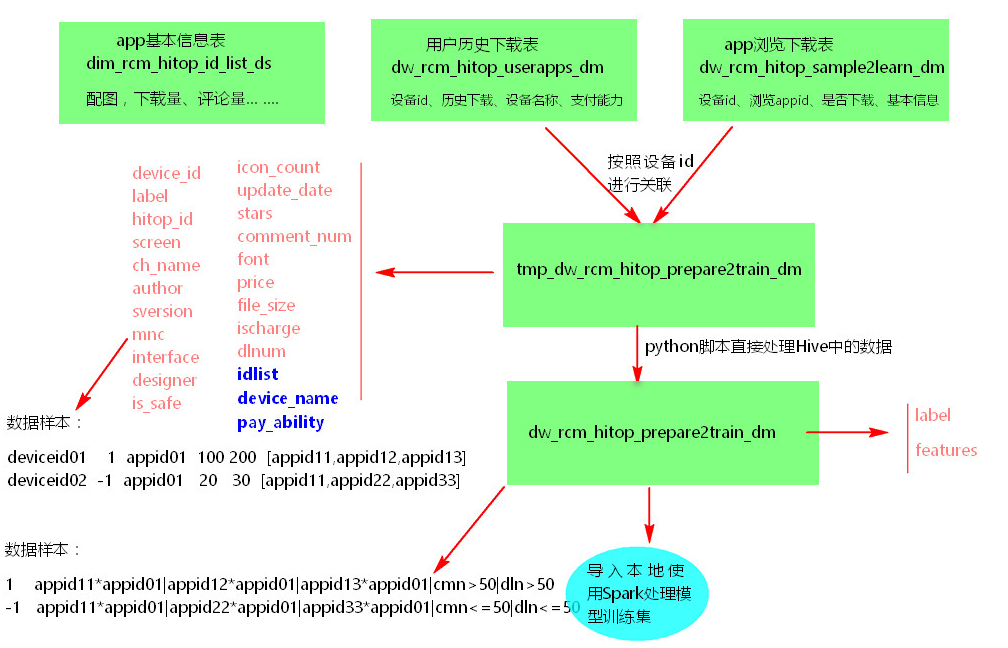
# 学习目标

|  |  |  |
| --- | --- | --- |
| 节数 | 知识点 | 要求 |
| 第一节 （推荐系统-数据处理流程） | 数据处理转换流程图 | 掌握 |
| 第二节（Hive构建训练数据） | Hive创建临时表 | 掌握 |
| Hive构建训练数据 | 掌握 |
| 第三节（python文件处理数据） | python文件预处理数据 | 掌握 |
| 导出数据 | 掌握 |
| 第四节（推荐系统-模型训练） | 推荐系统模型训练 | 掌握 |
| 推荐系统模型训练代码 | 掌握 |
| 将数据导入到Redis | 掌握 |
| 第五节（推荐系统-dubbo介绍） | dubbo介绍 | 掌握 |
| dubbo-demo案例 | 掌握 |
| 第六节（推荐系统-dubbo实现推荐服务） | dubbo实现推荐服务 | 掌握 |

# 推荐系统-数据处理流程

推荐系统数据处理首先是将Hive中的用户app历史下载表与app浏览信息表按照设备id进行关联，然后将关联数据使用python文件进行处理，将数据预处理为label和feature两列的临时数据，后期经过处理转换成逻辑回归 模型的训练集，进而得到模型文件。

数据处理流程图如下：



# Hive构建训练数据

1. **创建临时表**

创建处理数据时所需要的临时表

* 1. CREATE TABLE IF NOT EXISTS tmp\_dw\_rcm\_hitop\_prepare2train\_dm
  2. (
  3. device\_id STRING,
  4. label STRING,
  5. hitop\_id STRING,
  6. screen STRING,
  7. ch\_name STRING,
  8. author STRING,
  9. sversion STRING,
  10. mnc STRING,
  11. interface STRING,
  12. designer STRING,
  13. is\_safe INT,
  14. icon\_count INT,
  15. update\_date STRING,
  16. stars DOUBLE,
  17. comment\_num INT,
  18. font STRING,
  19. price INT,
  20. file\_size INT,
  21. ischarge SMALLINT,
  22. dlnum INT,
  23. idlist STRING,
  24. device\_name STRING,
  25. pay\_ability STRING
  26. )row format delimited fields terminated by '\t';

最终保存训练集的表

1. CREATE TABLE IF NOT EXISTS dw\_rcm\_hitop\_prepare2train\_dm
2. (
3. label STRING,
4. features STRING
5. )row format delimited fields terminated by '\t';
6. **训练数据预处理过程**

首先将数据从正负例样本和用户历史下载表数据加载到临时表中：

1. INSERT OVERWRITE TABLE tmp\_dw\_rcm\_hitop\_prepare2train\_dm
2. SELECT
3. t2.device\_id,
4. t2.label,
5. t2.hitop\_id,
6. t2.screen,
7. t2.ch\_name,
8. t2.author,
9. t2.sversion,
10. t2.mnc,
11. t2.interface,
12. t2.designer,
13. t2.is\_safe,
14. t2.icon\_count,
15. to\_date(t2.update\_time),
16. t2.stars,
17. t2.comment\_num,
18. t2.font,
19. t2.price,
20. t2.file\_size,
21. t2.ischarge,
22. t2.dlnum,
23. t1.devid\_applist,
24. t1.device\_name,
25. t1.pay\_ability
26. FROM
27. (
28. SELECT
29. device\_id,
30. devid\_applist,
31. device\_name,
32. pay\_ability
33. FROM
34. dw\_rcm\_hitop\_userapps\_dm
35. ) t1
36. RIGHT OUTER JOIN
37. (
38. SELECT
39. device\_id,
40. label,
41. hitop\_id,
42. screen,
43. ch\_name,
44. author,
45. sversion,
46. IF (mnc IN ('00','01','02','03','04','05','06','07'), mnc,'x') AS mnc,
47. interface,
48. designer,
49. is\_safe,
50. IF (icon\_count <= 5,icon\_count,6) AS icon\_count,
51. update\_time,
52. stars,
53. IF ( comment\_num IS NULL,0,
54. IF ( comment\_num <= 10,comment\_num,11)) AS comment\_num,
55. font,
56. price,
57. IF (file\_size <= 2\*1024\*1024,2,
58. IF (file\_size <= 4\*1024\*1024,4,
59. IF (file\_size <= 6\*1024\*1024,6,
60. IF (file\_size <= 8\*1024\*1024,8,
61. IF (file\_size <= 10\*1024\*1024,10,
62. IF (file\_size <= 12\*1024\*1024,12,
63. IF (file\_size <= 14\*1024\*1024,14,
64. IF (file\_size <= 16\*1024\*1024,16,
65. IF (file\_size <= 18\*1024\*1024,18,
66. IF (file\_size <= 20\*1024\*1024,20,21)))))))))) AS file\_size,
67. ischarge,
68. IF (dlnum IS NULL,0,
69. IF (dlnum <= 50,50,
70. IF (dlnum <= 100,100,
71. IF (dlnum <= 500,500,
72. IF (dlnum <= 1000,1000,
73. IF (dlnum <= 5000,5000,
74. IF (dlnum <= 10000,10000,
75. IF (dlnum <= 20000,20000,20001)))))))) AS dlnum
76. FROM
77. dw\_rcm\_hitop\_sample2learn\_dm
78. ) t2
79. ON (t1.device\_id = t2.device\_id);

# python文件处理数据

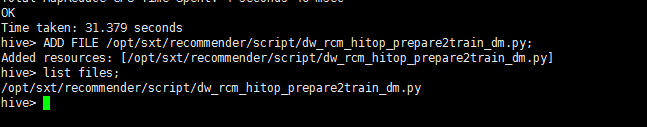
1. **python文件预处理数据**

针对Hive中“tmp\_dw\_rcm\_hitop\_prepare2train\_dm”数据，可以使用Hive自定义函数进行预处理，得到逻辑回归模型的训练集，这种方式需要编写代码，并且打包上传集群处理数据。这里，我们也可以在Hive中直接使用python对Hive中的数据进行预处理。

将python文件加载到Hive中：

1. ADD FILE /opt/sxt/recommender/script/dw\_rcm\_hitop\_prepare2train\_dm.py;

可以通过list files;查看是不是python文件加载到了hive：



在hive中使用python脚本处理数据的原理：Hive会以输出流的形式将数据交给python脚本，python脚本以输入流的形式来接受数据，接受来数据以后，在python中就可以一行行做一系列的数据处理，处理完毕后，又以输出流的形式交给Hive，交给了hive就说明了就处理后的数据成功保存到hive表中。

1. **python脚本内容**
   1. import sys
   2. if \_\_name\_\_ == "\_\_main\_\_":
   3. # random.seed(time.time())
   4. for l in sys.stdin:
   5. d = l.strip().split('\t')
   6. if len(d) != 21:
   7. continue
   8. # Extract data from the line
   9. label = d.pop(0)
   10. hitop\_id = d.pop(0)
   11. screen = d.pop(0)
   12. ch\_name = d.pop(0)
   13. author = d.pop(0)
   14. sversion = d.pop(0)
   15. mnc = d.pop(0)
   16. interface = d.pop(0)
   17. designer = d.pop(0)
   18. icon\_count = d.pop(0)
   19. update\_date = d.pop(0)
   20. stars = d.pop(0)
   21. comment\_num = d.pop(0)
   22. font = d.pop(0)
   23. price = d.pop(0)
   24. file\_size = d.pop(0)
   25. ischarge = d.pop(0)
   26. dlnum = d.pop(0)
   27. hitopids = d.pop(0)
   28. device\_name = d.pop(0)
   29. pay\_ability = d.pop(0)
   30. # Construct feature vector
   31. features = []
   32. features.append(("Item.id,%s" % hitop\_id, 1))
   33. features.append(("Item.screen,%s" % screen, 1))
   34. features.append(("Item.name,%s" % ch\_name, 1))
   35. features.append(("Item.author,%s" % author, 1))
   36. features.append(("Item.sversion,%s" % sversion, 1))
   37. features.append(("Item.network,%s" % mnc, 1))
   38. features.append(("Item.dgner,%s" % designer, 1))
   39. features.append(("Item.icount,%s" % icon\_count, 1))
   40. features.append(("Item.stars,%s" % stars, 1))
   41. features.append(("Item.comNum,%s" % comment\_num,1))
   42. features.append(("Item.font,%s" % font,1))
   43. features.append(("Item.price,%s" % price,1))
   44. features.append(("Item.fsize,%s" % file\_size,1))
   45. features.append(("Item.ischarge,%s" % ischarge,1))
   46. features.append(("Item.downNum,%s" % dlnum,1))
   47. #User.Item and User.Item\*Item
   48. idlist = hitopids.split(',')
   49. flag = 0;
   50. for id in idlist:
   51. features.append(("User.Item\*Item,%s" % id +'\*'+hitop\_id, 1))
   52. flag += 1
   53. if flag >= 3:
   54. break;
   55. # Output
   56. output = "%s\t%s" % (label, ";".join([ "%s:%d" % (f, v) for f, v in features ]))
   57. print(output)
2. **python预处理数据使用**
   1. INSERT OVERWRITE TABLE dw\_rcm\_hitop\_prepare2train\_dm
   2. SELECT
   3. TRANSFORM (t.\*)
   4. USING 'python dw\_rcm\_hitop\_prepare2train\_dm.py'
   5. AS (label,features)
   6. FROM
   7. (
   8. SELECT
   9. label,
   10. hitop\_id,
   11. screen,
   12. ch\_name,
   13. author,
   14. sversion,
   15. mnc,
   16. interface,
   17. designer,
   18. icon\_count,
   19. update\_date,
   20. stars,
   21. comment\_num,
   22. font,
   23. price,
   24. file\_size,
   25. ischarge,
   26. dlnum,
   27. idlist,
   28. device\_name,
   29. pay\_ability
   30. FROM
   31. tmp\_dw\_rcm\_hitop\_prepare2train\_dm
   32. ) t;
3. **导出数据**

将“dw\_rcm\_hitop\_prepare2train\_dm”表中的数据导入到本地处理，这里可以直接在集群中使用SparkMLlib直接处理，为了方便演示，将数据导入到本地处理。

* 1. insert overwrite local directory '/opt/data/traindata' row format delimited fields terminated by '\t' select \* from dw\_rcm\_hitop\_prepare2train\_dm;

注：这里是将数据导出到本地，方便后面再本地模式跑数据，导出模型数据。这里是方便演示真正的生产环境是直接用脚本提交spark任务，从Hive中获取数据经过Spark处理得到模型文件，将模型数据写往Redis中。

# 推荐系统-模型训练

1. **模型训练代码**

模型训练代码参照scala文件：Recommonder.scala

* 1. Logger.getLogger("org.apache.spark").setLevel(Level.ERROR)
  2. val conf = new SparkConf().setAppName("recommonder").setMaster("local[\*]")
  3. val sc = new SparkContext(conf)
  4. //加载数据，用\t分隔开
  5. val data: RDD[Array[String]] = sc.textFile("./traindata").map(\_.split("\t")).sample(false,0.1,100L)
  6. //得到第一列的值，也就是label
  7. val label: RDD[String] = data.map(\_(0))
  8. //sample这个RDD中保存的是每一条记录的特征名
  9. //-1 Item.id,hitop\_id53:1;Item.screen,screen6:1;Item.name,ch\_name80:1;Item.author,author1:1
  10. val sample: RDD[Array[String]] = data.map(\_(1)).map(x => {
  11. val arr: Array[String] = x.split(";").map(\_.split(":")(0))
  12. arr
  13. })
  14. //将所有元素压平，得到的是所有分特征，然后去重，最后索引化，也就是加上下标，最后转成map是为了后面查询用
  15. //dict 是所有数据的所有不重复的特征
  16. val allFeaturesMap: Map[String, Long] = sample.flatMap(x =>x).distinct().zipWithIndex().collectAsMap()
  17. //得到稀疏向量，为每条数据的features，与dict对比，缺少的特征补成0
  18. val sam: RDD[SparseVector] = sample.map((sampleFeatures :Array[String])=> {
  19. //index中保存的是，未来在构建训练集时，下面填1的索引号集合
  20. val currentOneInfoAllFeatureIndexs: Array[Int] = sampleFeatures.map(feature => {
  21. //get出来的元素程序认定可能为空，做一个类型匹配
  22. val currentFeatureIndex: Long = allFeaturesMap.get(feature).get
  23. //非零元素下标，转int符合SparseVector的构造函数
  24. currentFeatureIndex.toInt
  25. })
  26. //SparseVector创建一个向量
  27. new SparseVector(allFeaturesMap.size, currentOneInfoAllFeatureIndexs, Array.fill(currentOneInfoAllFeatureIndexs.length)(1.0))
  28. })
  29. //mllib中的逻辑回归只认1.0和0.0，这里进行一个匹配转换
  30. val trainData: RDD[LabeledPoint] = label.map(x => {
  31. x match {
  32. case "-1" => 0.0
  33. case "1" => 1.0
  34. }
  35. //标签组合向量得到labelPoint
  36. }).zip(sam).map(tuple => new LabeledPoint(tuple.\_1, tuple.\_2.toDense))
  37. //逻辑回归训练，两个参数，迭代次数和步长，生产常用调整参数
  38. val model = new LogisticRegressionWithLBFGS()
  39. .setNumClasses(2)
  40. .setIntercept(true)
  41. .run(trainData)
  42. //模型结果权重
  43. val weights: Array[Double] = model.weights.toArray
  44. //将map反转，weights相应下标的权重对应map里面相应下标的特征名
  45. val map: Map[Long, String] = allFeaturesMap.map(\_.swap)
  46. val pw = new PrintWriter("./model");
  47. for(i<- 0 until weights.length){
  48. //通过map得到每个下标相应的特征名
  49. val featureName = map.get(i)match {
  50. case Some(feature) => feature
  51. case None => ""
  52. }
  53. //特征名对应相应的权重
  54. val str = featureName+"\t" + weights(i)
  55. pw.write(str)
  56. pw.println()
  57. }
  58. pw.flush()
  59. pw.close()

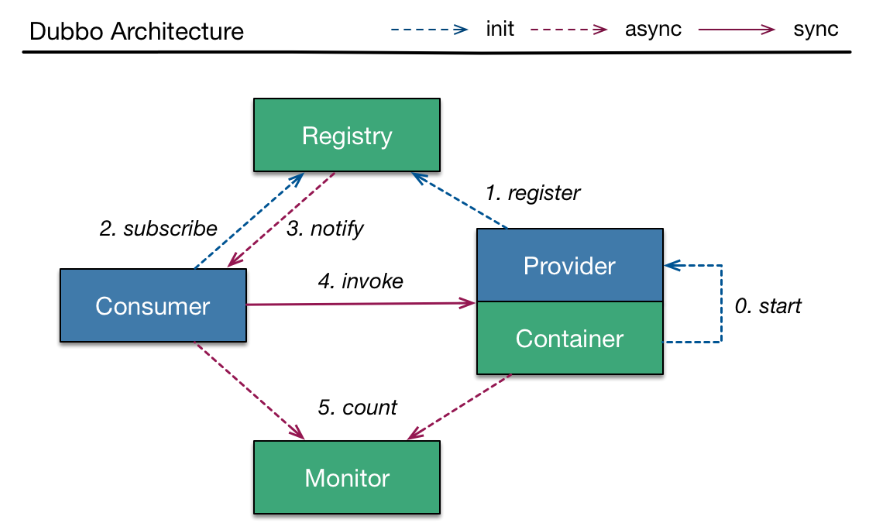
1. **将数据导入到Redis**

将app基本信息表、app历史下载表、app浏览下载表导入到Redis中，供后期dubbo推荐服务使用。

* 1. import redis
  2. # 将特征值模型文件数据存入redis数据库，将用户历史下载数据存入redis,将app基本描述商品词表存入redis数据库
  3. pool = redis.ConnectionPool(host='mynode4', port='6379', db=2)
  4. r = redis.Redis(connection\_pool=pool)
  5. f = open('./ModelFile.txt', "rb")
  6. while True:
  7. lines = f.readlines(100)
  8. if not lines:
  9. break
  10. for line in lines:
  11. kv = line.decode("utf-8").split('\t')
  12. r.hset("rcmd\_features\_score", kv[0], kv[1])
  13. f = open('./UserItemsHistory.txt', "rb")
  14. while True:
  15. lines = f.readlines(100)
  16. if not lines:
  17. break
  18. for line in lines:
  19. kv = line.decode("utf-8").split('\t')
  20. r.hset('rcmd\_user\_history', kv[0], kv[1])
  21. f = open('./ItemList.txt', "rb")
  22. while True:
  23. lines = f.readlines(100)
  24. if not lines:
  25. break
  26. for line in lines:
  27. kv = line.decode("utf-8").split('\t')
  28. # line[:-2] 取line 字符串的开头到倒数第二个的位置 数据，含头不含尾，也就是-2 就是将s 字符串中倒数后两个字符删除，常用在从文本读入数据的时候消除换行符的影响
  29. r.hset('rcmd\_item\_list', kv[0], line[:-2])
  30. print('all finished...')
  31. f.close()

# 推荐流程-dubbo介绍

dubbo介绍参照文档：dubbo.doc



# 推荐系统-dubbo实现推荐服务

利用dubbo实现推荐服务，核心代码如下：

1. public List<String> getRcmdList(String uid) {
2. // 获得数据库连接
3. Jedis jedis = new Jedis("mynode4", 6379);
4. jedis.select(2);
5. // 从用户历史下载表来获取最近下载
6. String downloads = jedis.hget("rcmd\_user\_history", uid);
7. String[] downloadList = downloads.split(",");
8. // 获取所有appID列表
9. Set<String> appList = jedis.hkeys("rcmd\_item\_list");
10. // 存储总的特征分值
11. Map<String, Double> scoresMap = new HashMap<String, Double>();
12. // 分别计算所有应用的总权重
13. for (String appId : appList) {
14. if(Arrays.asList(downloadList).contains(appId)) {
15. continue;
16. }
17. // 计算关联权重
18. double relativeFeatureScore = getRelativeFeatureScore(appId, downloadList, jedis);
19. updateScoresMap(scoresMap, appId, relativeFeatureScore);
20. // 计算基本权重
21. double basicFeatureScore = getBasicFeatureScore(appId, jedis);
22. updateScoresMap(scoresMap, appId, basicFeatureScore);
23. }
24. //这里将map.entrySet()转换成list
25. List<Map.Entry<String, Double>> list = new ArrayList<Map.Entry<String, Double>>(scoresMap.entrySet());
26. //然后通过比较器来实现排序
27. Collections.sort(list, new Comparator<Map.Entry<String, Double>>() {
28. //降序排序
29. public int compare(Map.Entry<String, Double> entry1,
30. Map.Entry<String, Double> entry2) {
31. return -entry1.getValue().compareTo(entry2.getValue());
32. }
33. });
34. // 打印分值
35. for (Map.Entry<String, Double> mapping : list) {
36. System.out.println(mapping.getKey() + ":" + mapping.getValue());
37. }
38. // 取前10个appID返回
39. List<String> result = new ArrayList<>();
40. int count = 0;
41. for (Map.Entry<String, Double> mapping : list) {
42. count++;
43. result.add(mapping.getKey());
44. if(count==10){
45. break;
46. }
47. }
48. jedis.close();
49. return result;
50. }

# 本节作业

1. 理解推荐系统处理数据流程。
2. 使用python处理Hive数据。
3. 使用dubbo实现服务。