基于半马尔科夫过程的买家作弊行为识别

爱俊 <Aijun Bai, aijunbai@gmail.com>

2015 年 3 月 6 号

背景

搜索行为质量中存在一些精刷的交易，比如买家利用协同行为、物流信息、资金关系和虚拟账号来帮助卖家达到刷单的目的，有的买家甚至模拟正常的购买行为来达到以假乱真的虚假成交，这批行为已经严重影响到淘宝和天猫交易平台的公正性以及电商市场的经营秩序。因此识别精刷行为对维护正常的电商市场环境起着至关重要的作用。精刷项目目标就是结合虚假交易来细化精刷的维度用来挖掘精刷背后的特点，从而帮助定位问题和处罚落地 [2]。

本工作关注于通过交易行为来识别作弊交易，进一步识别作弊买家。所谓交易行为是指买家购物时的状态序列，比如“搜索—点击商品—浏览商品详情页—浏览商品评论—创建订单—付款—成交”就是一条典型的购买行为路径，路径中每个节点就是一个状态。建模过程中使用的状态主要是从用户日志（Userlog）和交易表（tbods.s_tc_biz_order）中解析得到，所有遇到的状态数目有几百个。

如前所述，买家购买行为可以抽象出几个步骤，包括：搜索框输入、浏览搜索当前页、浏览搜索下一页，浏览搜索前一页、浏览商品概况、浏览商品详情、浏览商品评论、使用旺旺联系、创建订单、完成付款等。步骤的先后次序是不确定的，比如用户可能随机的搜索并浏览，但在概率意义上，是可以统计得到任意两个步骤之间的转移模型的，表现为一个带时间分布的转移矩阵。本工作的技术路线是，根据半马尔科夫过程过程（Semi-Markov Processes）分别建立作弊买家和正常买家的购物行为模型——即状态转移模型 [1]。完整的转移模型，形式上，包括：用户从状态 s_i 转移到 s_j 的概率 $\Pr(s_j \mid s_i)$，和该转移所需时间的概率分布 $f(t \mid s_i, s_j)$。分别建立好正常用户和作弊用户的模型之后，给定任意一笔交易的行为路径，就可以计算出该路径在正常模型和作弊模型中发生的概率，从而可以根据贝叶斯分类的原理给出该交易是作弊交易的概率。进一步，如果知道一个用户若干笔交易的

作弊情况，又可以估计出该用户是否经常参与作弊的概率。

本工作的基本假设是，不同类型用户在统计意义上的购物行为是不一样的，表现为最终建模得到的状态转移模型是不一样的。比如说，参与精刷的用户（或机器账号）没有实际购物的意图，仅仅是为了完成刷单的任务而“假意”完成购物行为。这类用户在购物时，每一步状态转移跟正常用户可能都是不一样的，即使某些步骤的转移已经伪装的很像了，也很难在概率意义上做到整个路径都伪装的跟正常用户完全一致——因为作弊用户是无法知道正常用户状态转移的概率模型的。如果已知一批买家的白名单和黑名单，就可以分别建模好正常买家和作弊买家的模型后，就可以进一步使用贝叶斯方法给出任意一个买家某次购买行为有问题的概率，从而给出该买家是否经常作弊的概率。

方法

本节介绍所提方法的主要内容，包括：状态定义、训练样本、模型训练、贝叶斯分类和用户判定。

状态定义

目前，状态的定义主要有两个来源：用户日志和交易表。用户日志数据给出了用户在搜索和详情页浏览是的埋点信息；交易表给出了部分支付和支付后阶段的状态信息。

挂川整理的用户日志表（boyang_wormhole_user_log）记录了每个埋点数据对应的机器 ID一终端类型（mid_terminal）、用户 ID（uid）、动作类型（action）、动作位置（position）、时间（time）、附加信息（info）等字段，为定义搜索和详情页浏览的状态节点提供了极大便利。mid_terminal 给出了机器的 ID 以及终端类型（pc 或 wireless）。uid 给出了对应用户的 ID。time 给出了记录该埋点数据的时间（格式形如“135959”表示 13 点 59 分 59 秒）。action 的可能取值包括：search、click 和 trade。action 取 search 时，position 给出搜索行为的几种类型：tb_sc_S（淘宝搜索）和 tm_sc（天猫搜索），info 给出搜索的排序方法、query 内容和当前浏览的结果页数（以 0x001 分隔）；action 取 click 时，position 给出用户在详情页内点击的位置，比如：2013.1.1000126.27（信用支付）、2013.1.1000126.21（卖家名片 _ 进入店铺）、/tbdetail.12.6（官方浮动条 _ 成交记录）等，info 不提供更多信息；action 取 trade 时，position 取值为 trad，info 给出对应交易的订单号。一般情况下，action 和 position 一起给出一个具体状态，命名为 action:position。需要注意的是，如果需要详细描述搜索行为，还应该把 info 信息加上，但

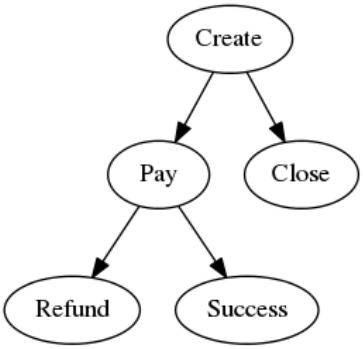

图 1: 支付状态转移图

这不还不是本工作目前阶段的重点。

交易表(tbods.s_tc_biz_order)给出了每笔交易的订单创建时间(gmt_create)、订单支付时间（pay_time)、订单关闭时间（end_time）和订单支付状态(pay_status)。这里感兴趣的支付状态是：4（退款）, 6（成交）和 8（交易关闭)。因此，可以抽象出 3 条状态状态转移的路径，见图 1。其中，Create 为订单创建，Play 为订单支付，Close 为订单取消（或关闭），Refund 为订单退款成功而结束，Success 为订单成功结束。Create 状态停留一段时间以后，可能转移到 Pay 状态或 Close 状态。Pay 状态停留一段时间以后，可能转移到 Refund 状态或 Success 状态。注意到，这个过程里面不涉及到状态的前向跳转。

结合 boyang_wormhole_user_log 表和 tbods.s_tc_biz_order 表，根据交易的订单号，就可以解析出来一笔完成订单的完整状态转移路径和状态转移的时间样本。

训练样本

本文所提方法要求事先给出作弊用户和正常用户的订单样本。目前主要包括以下几类样本:

1. Apass 用户订单：用户名单来自挂川的 boyang_buyer_white_list 表，本文认为 Apass 用户的所有订单都是非作弊订单

2. 垃圾账号订单：原始垃圾账号名单由网安提供，原始数据表已过期。一份拷贝的数据位于 t_baj_laji_users 表。本文认为垃圾账号的所有订单都是作弊订单

3. 物流作弊订单:来自天栋的 antispam_wl_spam_js_black_wl_info_d 表 bid 字段

4. 炒信作弊订单：来自问长的 copy_alipay_to_etao_sim_network_d 表 id_2 字段

训练时，根据二元分类模型 (Binary Classification)，把垃圾账号、物流作弊订单和炒信作弊订单看成负样本；把 Apass 用户订单看成正样本。尝试过多元分类的模型 (Multiclass Classification)，但具体每个作弊类型的准确率和召回率不及二元分类。多元分类可以作为下一步工作的一个尝试方向。

模型训练

假设已知一条状态转移路径：$\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, \ldots, \langle s_N, t_N\rangle$，表示用户在 t_i 时刻进入状态 s_i。那么，可以按照如下规则记录 s_i 到 s_j 的转移次数和转移时间样本：$T[s_i, s_j] = T[s_i, s_j] + 1$，和 $D[s_i, s_j] = D[s_i, s_j] \cup t_j - t_i$，其中，$T[s_i, s_j]$ 给出观察到的 s_i 到 s_j 转移次数，$D[s_i, s_j]$ 给出 s_i 到 s_j 转移的所有时间样本集合。统计一类用户所有订单的状态转移路径，得到最终的 T 矩阵和 D（集合）矩阵，就可以估计出该半马尔科夫过程的转移模型：

$$\Pr(s_j \mid s_i) = \frac{T[s_i, s_j]}{\sum_s T[s_i, s]}, \tag{1}$$

和

$$f(t \mid s_i, s_j) = \tilde{\Pr}(t \mid D[s_i, s_j]), \tag{2}$$

其中 $\tilde{\Pr}$ 表示某种概率估计方法。目前使用的是滑动平均 (Moving Average) 方法，后续可以试验更多其他方法，比如核密度估计方法（Kernel Density Estimation）等。

贝叶斯分类

假设训练得到的一个半马尔科夫模型为 M，对一条状态转移路径 $p = \{\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, \ldots, \langle s_N, t_N\rangle\}$，可以计算出该路径在该模型中发生的概率为：

$$\Pr(p \mid M) = \prod_{i \in [1, N-1]} \Pr_M(s_{i+1} \mid s_i) f_M(t_{i+1} - t_i \mid s_i, s_{i+1}), \tag{3}$$

其中 $\Pr_M$ 和 f_M 给出模型 M 的状态转移概率和转移时间分布。

令 M' 和 M'' 分别表示训练得到的正常用户和作弊用户的状态转移模型，则路径 p 在这两个模型内发生的概率分别为：$\Pr(p \mid M')$ 和 $\Pr(p \mid M'')$，所以，根据贝叶斯原理，该交易路径作弊的概率为：

$$\Pr(M'' \mid p) = \frac{\Pr(p \mid M'') \Pr(M'')}{\Pr(p \mid M'') \Pr(M'') + \Pr(p \mid M') \Pr(M')}, \tag{4}$$

	abnormal_pc	normal_pc	Samples
abnormal_pc	10829	1191	12020
normal_pc	1934	17462	19396
Outcomes	12763	18653	31416

表 1: 决策阈值 $\gamma = 0.5$ 时的 pc 端交叉验证结果

其中，$\Pr(M')$ 和 $\Pr(M'')$ 分布为正常用户和作弊用户的先验概率。目前的实现中，认为 $\Pr(M') = \Pr(M'') = 0.5$。令 γ 为作弊行为的决策阈值，如果 $\Pr(M'' \mid p) > \gamma$，就认为该交易是作弊的。

用户判定

给定决策阈值 γ，如果一个用户有 n 笔作弊订单，m 笔正常订单，就认为该用户有 $\frac{n+1}{m+n+2}$ 的概率作弊。后续可以尝试其他方法。

实验验证

由于时间和资源的限制，目前尚未完成产品部门的测试，但对算法的分类结果进行了交叉验证。具体地，取垃圾账号订单、物流作弊订单和资金炒信订单的 99% 的数据建立非正常买家购物行为的半马尔科夫模型；取剩下的 1% 数据（pc 端共涉及 31416 笔订单数据）作为测试集测试产出模型在不同决策阈值下的召回率和准确率。决策阈值 γ 表示作弊的概率大于 γ 就认为该笔交易作弊。

以 pc 端为例，表 1和表 2分别给出了 γ 取 0.5 和 0.86 时的测试结果，其中 abnormal_pc、normal_pc 和 recognized 分别表示：pc 端作弊、pc 端正常和无法识别。无法识别是指，$\Pr(M'' \mid p) \leq \gamma$ 并且 $\Pr(M' \mid p) \leq \gamma$ 的情况。$\gamma = 0.5$ 时，pc 端作弊行为对应的召回率为 0.90，准确率为 0.85；$\gamma = 0.86$ 时，pc 端作弊行为对应的召回率为 0.86，准确率为 0.91。结果显示产出模型在测试集上针对 pc 端作弊行为的召回率和准确率是比较高。实际应用时，可以通过调整决策阈值 γ 在召回率和准确率之间进行平衡。

	abnormal_pc	normal_pc	unrecognized	Samples
abnormal_pc	10366	741	913	12020
normal_pc	1005	15013	3378	19396
Outcomes	11371	15754	20321	31416

表 2: 决策阈值 $\gamma = 0.86$ 时的 pc 端交叉验证结果

代码

截至本文档完成之日，完整的代码位于团队 SVN 目录：http://svn.develop.taobao.net/repos/searchAppCodes/AntiSpam_Tsc/trunk/aijun.baj/buyer_model-userlog。本节主要介绍现有代码的主要结构、实现细节和运行示例。

目录结构

目前的主要目录结构如下：

conf/: 一些配置文件

data/: 运行过程中的一些数据文件

doc/: 相关文档

py/: Python 源文件

sh/: Shell 源文件

sql/: 运行过程中的临时 Sql 脚本

classify.sh: 离线分类脚本

evaluate.sh: 离线交叉验证脚本

run.sh: 在线训练、分类脚本

train.sh: 在线分类脚本

test.sh: 离线测试脚本

resources.zip: Python 源文件压缩包

models.zip: 训练得到的模型文件压缩包

README.txt: 说明文件

实现细节

一些值得指出的实现细节如下:

- sh/merge_history.sh 脚本通过 mid_terminal 和 uid 拼接用户日志，每个拼接后的日志都追加了一个 bid 字段。由于 click 和 search 动作不包含 bid 信息，只有 trade 动作有 bid 信息，每个拼接后的日志实际追加了所有可能匹配的 bid 信息，输出为多条记录。py/history.py 脚本中通过 bid 对用户日志进行了分段（split）和挑选（select）操作。
- Create、Play、Close、Refund 和 Success 为根据交易表抽象出来的状态，不直接存在于用户日志埋点数据中，代码中对它们分别构建了虚拟埋点数据（py/state.py 文件中），以便可以跟其他埋点数据一起统一处理。
- py/state.py 文件中，还构建了虚拟的开始状态（Start）和结束状态（Finish），py/history.py 输出的每条分段后的路径都补全了开始和结束状态。
- 在线训练得到的模型，通过序列化（pickle）、压缩（bz2）、Base64 编码（base64）和分段（chunk）后直接按照 ODPS 表格的形式存储，每条记录最大为 8M。
- py/merger.py 文件的作用是增加 Mapp-Reduce 的层次数，以提高 ODPS 线上运行的并行率。
- resources.zip 的作用是使得 ODPS Streaming 运行时可以更方便的 import 自定义 Python 模块。
- models.zip 的作用是通过 ODPS 资源文件的形式上传训练得到的模型文件，供在线分类使用。
- py/semi_markov.py 文件实现了所用到的半马尔科夫过程的核心数据结构和算法，Decimal 的使用是为了实现高精度浮点数计算。

运行示例

一些脚本调用示例如下:

./run.sh -c -s -h: 预处理样本数据、解析行为历史数据、在线训练模型、在线分类并离线交叉验证。由于一些样本数据可能不全, 所以需要慎重删除过去处理好的样本数据。在线分类得到的输出是 t_baj_user_log_merged_classified

表和 t_baj_user_classified 表。t_baj_user_log_merged_classified 表给出订单维度的作弊识别数据，t_baj_user_classified 表给出用户维度的作弊识别数据。

./test.sh -n 100 -r train.sh: 通过 dship 下载解析后的用户交易历史数据(t_baj_user_log_merged)到本地（data.txt），并离线测试 train.sh 脚本。dship 可能会下载过多文件，所以必要时需要手动通过 Ctrl+C 结束 dship 下载过程。

./test.sh -n 100 -r classify.sh: 通过 dship 下载解析后的用户交易历史数据（t_baj_user_log_merged）到本地（data.txt），并离线测试 classify.sh 脚本。dship 可能会下载过多文件，所以必要时需要手动通过 Ctrl+C 结束 dship 下载过程。

总结

本文给出了基于半马尔科夫过程的买家用户作弊行为识别方法，并介绍了相关代码实现和运行示例。交叉验证的结果显示，该方法的识别召回率和准确率较高，表明概率意义上理解和建模用户行为是可行的。该思路具有较好的应用前景，后续可以尝试扩展本方法到更多的应用场景。

参考文献

[1] Guédon, Y.: Semi-Markov Chains and Hidden Semi-Markov Models toward Applications: Their Use in Reliability and DNA Analysis by BARBU, V. S. and LIMNIOS, N. Biometrics 65, 1312–1312 (2009)

[2] 行为质量团队: 行为质量之反精刷. 淘宝网搜索事业部 Wiki: http://searchwiki.taobao.ali.com/index.php/%E8%A1%8C%E4%B8%BA%E8%B4%A8%E9%87%8F%E4%B9%8B%E5%8F%8D%E7%B2%BE%E5%88%B7#.E8.BF.9B.E5.B1.95 (2014)