# 电视广告分户推送与营运规划
# ——2019 年深圳杯 D 题

## 摘要

电视台通过向广告主销售广告时段获取销售利益. 因此, 卖方确定的合理广告出售底价以及买方确定的合理广告费支出对于买卖双方利益尤为重要. 除此以外, 实现广告与用户的匹配推送对于提升买方收益以及卖方业务利润均有显著作用. 本文着重对这一问题进行深入的分析以及建模.

对于问题一, 我们通过对广告以及用户特征向量的筛选, 选择用户年龄段, 用户性别, 用户教育水平, 用户所在行业, 用户消费水平以及用户的婚姻状态等作为用户的标签输入, 将广告中产品所属类型（即广告类型）, 广告中产品价格, 作为广告的标签输入, 结合因子分解机和神经网络, 搭建多标签模型, 通过训练数据, 得出静态情况下的" 广告 -用户" 分类匹配推送模型, 即输入一个用户和所有广告所拥有的标签, 遍历所有类型广告, 得到用户对于所有广告的匹配值, 并从中找到最为匹配的一则广告进行分户推送.

对于问题二, 我们结合实际选取与广告价格有关的数据量, 即节目类型, 广告播放时段, 直播关注度, 市占率, 时间等. 运用多元线性回归的方法, 确定分时段广告价格为被解释变量, 上述提到的变量为解释变量. 我们决定将回归过程分解为三阶段, 即根据节目类型以及时段预测直播关注度, 根据直播关注度以及节目类型预测市占率, 根据市占率, 节目类型, 广告位置以及时段预测解最终的合理估算底价. 同时在静态合理底价估算模型基础上, 运用时间序列模型对收视率实现预测, 将静态模型动态化, 实现问题二中的动态合理低价估算模型. 经实证检验, 我们的回归模型显著, 对底价的预测能力较好, 且 ARMA 时间序列模型平稳.

对于问题三, 在问题一的静态模型的基础之上, 将历史收视纪录、用户画像特征和当前播放广告的销售量作为输入. 分别建立基于历史记录和协同过滤的推荐模型、基于用户特征的推荐模型以及基于在播视频广告的产品销售情况的三个子模型, 并且根据用户的历史记录总数计算置信度, 决定三个子模型在最终推荐结果之中的权重占比, 实现" 用户 -广告" 分户匹配推送. 同时, 根据用户历史记录随着时间而产生的在内容与数目上的变化, 实时对推荐模型进行更新, 建立" 频道用户 -视频广告" 分类匹配推送更新模型.

对于问题四, 我们首先建立静态的拍卖模型, 我们构造了买方的收入函数, 将求解合理广告费的问题转化为求解极大值点的问题. 当投入广告带来的效益大于不投入的效益与机会成本之和时, 该厂商参与竞拍, 且根据用户接受度选择最合适的电视台投放广告. 为了保证卖方的利益采用第二密封竞价模型. 然后建立动态的模型, 引入时间变量, 用 $t-1$ 时的销量等参数推断 $t$ 时的参与广告拍卖的厂家合理的广告投入, 形成一个不完全信息动态博弈过程. 将原本复杂的竞价交易模型简化为通俗易懂的数学模型. 实证表明, 我们的模型能客观的反映出优胜劣汰的交易市场规律.

**关键字：** BP 神经网络, 供求关系, 多元线性回归, ARMA 时间序列模型, 重复博弈, 不完全信息动态博弈, 第二密封竞价模型

# 目录

# 一、 问题重述

我们需要解决的问题如下:

1. 通过对于视频广告特征的分析, 选取合适的特征变量, 同时选取对应电视台用户画像当中适当的特征值, 建立数学模型, 研究二者之间的关系, 实现静态下的数据匹配.
2. 通过对分时段底价的影响因素分析, 选择合理的影响因素, 建立数学模型, 来估算卖方分时段竞卖的底价.
3. 研究电视频道用户的收视情况以及在播广告的产品销售情况, 基于问题一当中的静态模型, 实现电视台视频广告及用户的动态匹配推送模型, 实现广告推送业务的更新化, 与此同时, 结合模型, 预测出产品未来销售情况变化.
4. 在第三题的模型的基础上, 建立数学模型, 模拟竞价交易的过程, 同时实现卖方收益的最大化, 并尽可能提升收视率及买方的产品销售量以及利润.
5. 设计求解算法, 同时结合现有数据, 验证问题一至四中的模型.

# 二、 建模分析

对于问题一, 我们通过对广告以及用户特征向量的筛选, 选择用户年龄段, 用户性别, 用户教育水平, 用户所在行业, 用户消费水平以及用户的婚姻状态等作为用户的标签输入, 将广告中产品所属类型（即广告类型）, 广告中产品价格, 作为广告的标签输入, 结合因子分解机和神经网络, 搭建多标签模型, 通过训练数据, 得出静态情况下的" 广告 -用户" 分类匹配推送模型, 即输入一个用户和所有广告所拥有的标签, 遍历所有类型广告, 得到用户对于所有广告的匹配值, 并从中找到最为匹配的一则广告进行分户推送.

对于问题二, 我们结合实际选取与广告价格有关的数据量, 即节目类型, 广告播放时段, 直播关注度, 市占率, 时间等. 运用多元线性回归的方法, 确定分时段广告价格为被解释变量, 上述提到的变量为解释变量. 将预处理过的数据输入 Eviews, 测算变量之间相关性. 经过 T 检验, 决定将回归过程分解为三阶段, 即根据节目类型以及时段预测直播关注度, 根据直播关注度以及节目类型预测市占率, 根据市占率, 节目类型, 广告位置以及时段预测解最终的合理估算底价. 同时在静态合理底价估算模型基础上, 运用时间序列对收视率实现预测, 将静态模型动态化, 通过 ARMA 模型拟合, 确定时间序列模型形式, 最终实现模型动态化.

对于问题三, 在问题一的静态模型的基础之上, 将历史收视纪录、用户画像特征和当前播放广告的销售量作为输入. 分别建立基于历史记录和协同过滤的推荐模型、基于用户特征的推荐模型以及基于在播视频广告的产品销售情况的三个子模型, 并且根据用户的历史记录总数计算置信度, 决定三个子模型在最终推荐结果之中的权重占比, 实现" 用户 -广告" 分户匹配推送. 同时, 根据用户历史记录随着时间而产生的在内容与数目上

的变化, 实时对推荐模型进行更新, 建立” 频道用户 -视频广告” 分类匹配推送更新模型.

对于问题四, 我们首先建立静态的拍卖模型, 我们先基于广告费用, 竞争对手数, 最近一个生产周期的销量, 构造买方的收入函数, 将求解合理广告费的问题转化为求解极大化问题. 当投入广告带来的效益大于不投入的效益与机会成本之和时, 该厂商参与竞拍, 且根据用户接受度选择合适的电视台投放广告. 为了保证卖方的利益, 我们采用第二密封竞价模型. 然后建立动态的模型, 引入时间变量, 用 $t-1$ 时的销量等参数推断 $t$ 时的各个参数, 形成一个不完全信息动态博弈过程, 由博弈论理论知 [1], 这个过程的均衡一定存在.

对于问题五, 我们对部分题目给出了算例.

# 三、 模型的假设

- 各频道在一段时间内的节目安排不变, 变化的只是它们之间的广告;
- 市场出清, 即商品价格具有充分的灵活性, 能使供给和需求迅速达到均衡.
- 所有人都是经济人, 具有完全理性.
- 市场是完全竞争市场, 厂商之间信息是不完全的.
- 各厂商的销售函数 $sell^i$、收入函数 $f^i$、机会成本函数 $\pi^i$ 在一段时间内形式不变.
- 市场是有限的, 因此收益函数取值不可能为 $+\infty$.
- 具有相似显性特征的用户在某种程度上具有相似的兴趣.

# 四、 变量及符号说明

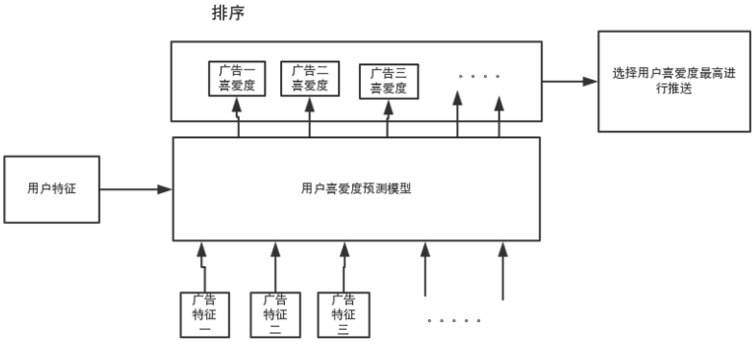
| 符号 | 意义 |
|---|---|
| $D_{1i}$ | 虚拟变量, 用于标记不同的节目类型 |
| $D_{2i}$ | 虚拟变量, 用于标记广告播放时段 |
| $T$ | 节目开始时间的相对值 |
| $focus$ | 直播关注度 |
| $rate$ | 市占率 |
| $A_t^i$ | 时刻 $t$ 第 $i$ 个参与拍卖的厂商 |
| $n_t$ | 时刻 $t$ 某电视台投放广告的同一行业厂商数 |
| $N_t^*$ | 时刻 $t$ 市场上某行业总厂商数 |
| $N_t$ | 时刻 $t$ 参与竞拍的厂商总数 |
| $sell^i$ | 厂商 $A^i$ 的销售函数 |
| $VC^i$ | 厂商 $A^i$ 的可变成本 |
| $FC^i$ | 厂商 $A^i$ 的不变成本 |
| $P_t^i$ | 厂商 $A^i$ 为使利益最大化而投入的广告费 |
| $P_t^0$ | 时刻 $t$ 卖方的底价 |
| $I(A)$ | 示性函数, $A$ 为 true 是 $I(A)=1$ 否则为 0 |
| $Ca_i$ | 绝对置信度 |
| BSE | 协同过滤的排序依据 |
| $P_H$ | Eploitation 的匹配向量 |
| $Pcf$ | 基于协同过滤的匹配向量 |
| $P_i$ | 基于子模型 i 的匹配向量 |
| $W_i$ | 三种子模型的权重 |
| $K$ | 协同过滤中, 选择的 Top-K 近似的用户 |

# 五、 模型建立与求解

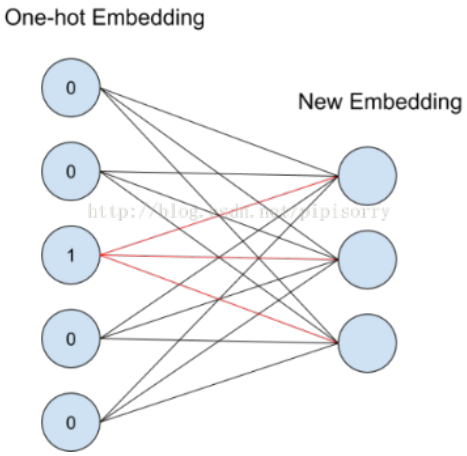
## 5.1 问题一

### 5.1.1 广告——用户分类匹配推送静态分类模型

我们通过对广告特征以及用户画像的分析, 分别选取出了针对于广告特征以及用户画像的特征变量, 用户教育水平, 用户所在行业, 用户消费水平以及用户的婚姻状态等作为用户的标签输入, 将广告中产品所属类型（即广告类型）, 广告中产品价格, 作为广告的标签输入. 最后遍历所有类型广告, 得到用户对于所有广告的匹配值, 并从中找到最为匹配的一则广告进行分户推送.




首先, 用 python 的 selenium( 浏览器框架测试库), 我们从酷云平台上导出了含有用户画像和广告内容匹配信息的数据. 并对其进行了数据清洗, 得到上述的特征, 并进行了 one-hot 编码. 然而, 此时数据特征过于稀疏且浪费计算资源, 为此我们加入了 embedding 层以对原始数据进行降维. 嵌入层 (embedding layer) 的结构如下图所示.



通过分析原始数据后, 我们发现数据之间的相关性较为复杂, 此时, 传统的特征工程的局限性在于, 仅仅通过人为的观察和归纳, 难以得到更高阶的特征. 然而, 通过因子分解机 (Factorization Machines, FM) 可以做到利用一阶特征和二阶特征的组合, 因为引入了隐变量的原因, 对于几乎不出现或者很少出现的隐变量, FM 也可以很好的学习对用户与广告的匹配值进行回归. FM 在基本线性回归模型的基础上引入交叉项, 如下

$$y_{FM} =< w, x > + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} < V_i, V_j > x_{j_1} x_{j_2}$$

若是这种直接在交叉项 $x_i x_j$ 的前面加上交叉项系数 $w_{ij}$ 的方式在稀疏数据的情况下存在一个很大的缺陷, 即在对于观察样本中未出现交互的特征分量, 不能对相应的参数进行估计. FM 对每一个特征分量 $x_i$ 引入辅助向量 $v_i = (v_{i1}, v_{i2}, \cdots, v_{ik})$, 利用 $v_i v_j^T$ 对交叉项的系数 $w_{ij}$ 进行估计, 即$\hat{w}_{ij} = v_i v_j^T$, 令

$$V = \begin{pmatrix} v_{11}, v_{12}, \cdots, v_{1k} \\ v_{21}, v_{22}, \cdots, v_{2k} \\ \vdots, \vdots, \cdots, \vdots \\ v_{n1}, v_{n2}, \cdots, v_{nk} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

则 $\hat{w} = VV^T$, 用这种方式可以有效地对二阶的特征组合进行回归.

对于预估用户与广告的匹配值的问题, 最重要的是学习到用户的特征与广告的特征背后隐含的特征组合. 在不同的推荐场景中, 低阶组合特征或者高阶组合特征都会对最终匹配值产生影响.

之前建立的因子分解机 (Factorization Machines, FM) 通过对于每一维特征的隐变量内积来提取特征组合, 仅能做到对一阶特征和二阶组合特征进行建模. 于是为了解决更高阶的特征组合的问题, 我们通过多层的神经网络即 DNN 去解决.

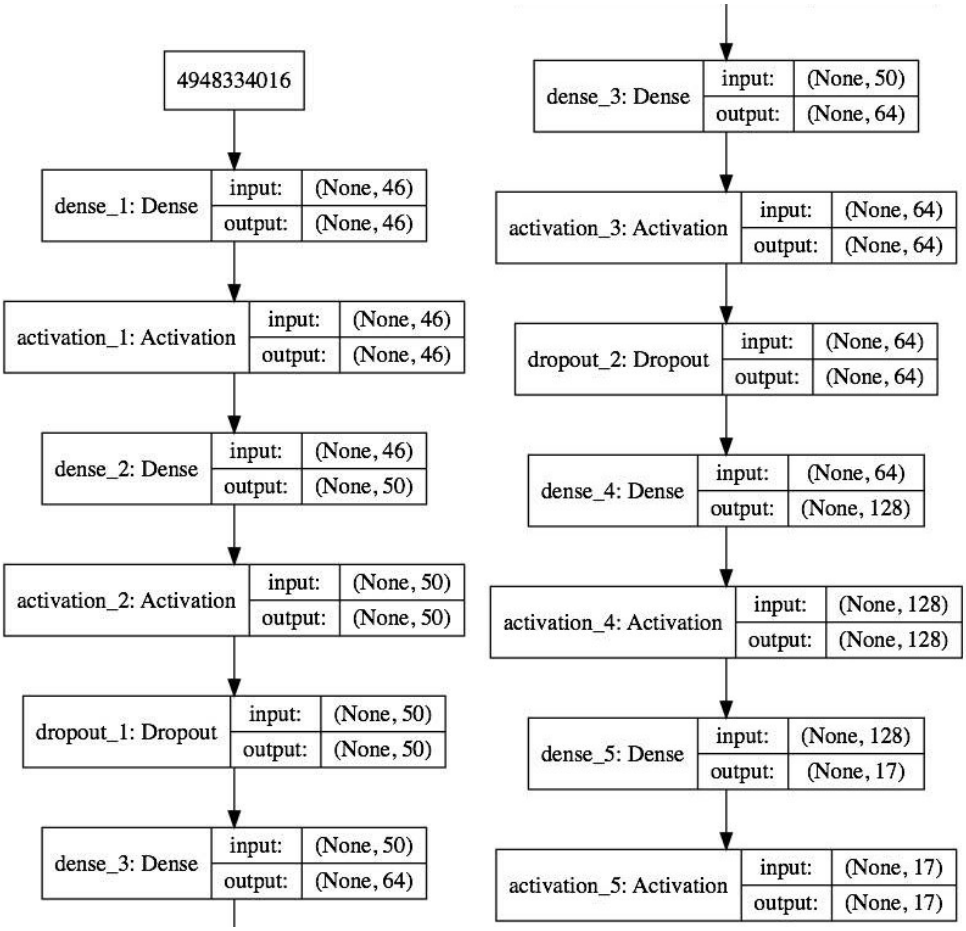
运用基于 tensorflow 的 pytorch 深度学习库, 搭建了一个前馈神经网络. 其层与层之间的抽象关系如图 1所示:

有一个输入层 dense1, 一个输出层 activation_5, 以及 10 个隐藏层. 其中的 activation 层, 起到激活函数的作用, 在神经网络中进行非线性变换. 其中的 dense 层, 与上一层所有的 cell 相连, 对前一层的特征求加权和, 起到分类器的作用. 每一组 dense 层和 activation 层神经元的功能如图 2所示:

从左向右看, $x_i$ 表示由上一个神经元传入的数值. 权重 $\omega$ 中, $i$ 表示前一层神经元的序号, $j$ 表示指向的神经元的序号, 上标 $l$ 为层次. 偏置 $b_j$ 是属于这个神经元的, $j$ 表示这个神经元的序号. 经加权求和后得到 $v_j$, 其中的 $M$ 为前一层的神经元总数. 再将结果代入激活函数 $\varphi$ 中, 得到 $y_j$ 作为此神经元的输出.

具体而言, 输入层函数为

$$v_j = \sum_{i=1}^{M} \omega_{ij} x_i + b_j$$




(a) 静态分类神经网络 1 至 7 层　　(b) 静态分类神经网络 7 至 13 层

**图 1　静态分类神经网络**

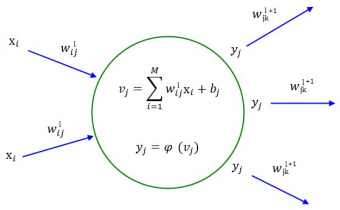


**图 2　神经元**

隐藏层所采用的激活函数为 relu 函数

$$\varphi(x) = \max\{0, x\}$$

输出层函数为

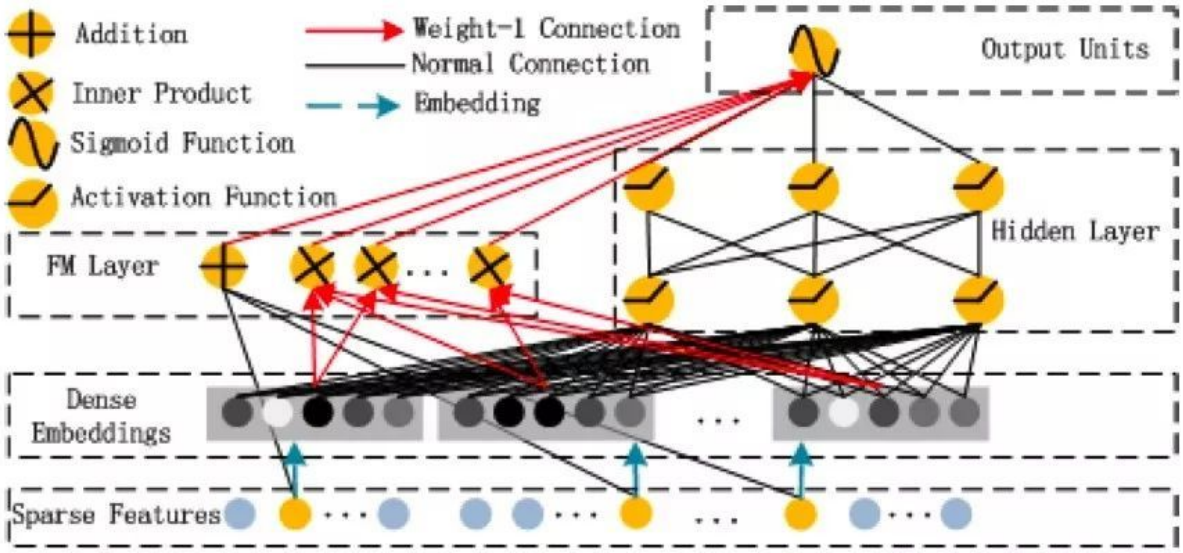
$$y_j = \varphi(v_j)$$

利用 DNN 进行建模虽然可以解决高阶特征组合的问题, 但低阶和高阶特征组合隐含地体现在隐藏层中, 我们希望把低阶特征组合单独建模, 然后融合高阶特征组合, 即将 DNN 与 FM 进行一个合理的融合. 由此, 我们建立了我们最终的模型 DeepFm.

DeepFM 包含两部分: 神经网络部分与因子分解机部分, 分别负责低阶特征的提取和高阶特征的提取. 这两部分共享同样的输入. 输出层的公式为

$$\hat{y} = sigmoid(y_{FM} + y_{DNN})$$

以及模型的示意图如图所示:




## 5.2 问题二

### 5.2.1 静态合理底价估算模型

对于问题二, 设某一频道的节目类型有 $n+1$ 种, 对应 $n$ 个虚拟变量 $D_{1i}$, $i = 1, 2, \cdots, n$, 对应第 $j$ 种节目类型, 虚拟变量的取值为 $D_{1i} = \delta_{ij}$, $\delta_{ij}$ 为 Kronecker 符号. 广告播放时段分为节目前, 节目中插播, 节目结束后, 对应虚拟变量 $D_{21}$, $D_{22}$, 取之方法与上述同理. 设节目开始时间的相对值为 $T$, $T \in [0, 1)$, 比如晚上 7 点开始播放的新闻联播, 其 $T = \dfrac{19}{24}$.

于是, 我们可以定义合理底价估算函数 $p(D_{11}, D_{12}, \cdots, D_{1n}, D_{21}, D_{22}, T)$, 这个函数可分解为以下函数:

- 直播关注度 $focus$ 函数: 由于直播关注度和广告的位置无关, 故自变量选取应该去掉 $D_{21}$, $D_{22}$; 得到如下函数形式:

$$focus = focus(D_{11}, D_{12}, \cdots, D_{1n}, T)$$

- 市占率 $rate$ 函数: 同样的市占率和广告的位置无关, 故自变量选取应该去掉 $D_{21}$, $D_{22}$; 得到如下函数形式:

$$rate = rate(focus, D_{11}, D_{12}, \cdots, D_{1n}, T)$$

- 合理底价估算函数:

$$p = p(rate, D_{11}, D_{12}, \cdots, D_{1n}, D_{21}, D_{22}, T)$$

我们用多元线性回归模型来给出这三个函数的估计形式:

$$\begin{aligned} focus =& C + \sum_{i=1}^{n} \alpha_{1i} D_{1i}(\text{虚拟变量影响截距项}) \\ &+ f(T) + \sum_{i=1}^{n} \alpha_{2i} D_{1i} f(T)(\text{虚拟变量影响斜率}) \\ &+ \mu(\text{随机扰动项, 假设服从 } N(0, \sigma_1^2)) \\ rate =& C + \beta_1 focus + \sum_{i=1}^{n} \beta_{1i} D_{1i} + \sum_{i=1}^{n} \beta_{2i} D_{1i} focus + f(T) + \mu \\ p =& C + \gamma_1 rate + \sum_{i=1}^{n} \gamma_{1i} D_{1i} + \sum_{i=1}^{n} \gamma_{2i} D_{1i} rate + f(T) \\ &+ \gamma_{31} D_{21} + \gamma_{32} D_{22} + \gamma_{41} D_{21} rate + \gamma_{42} D_{22} rate + \mu \end{aligned}$$

这里 $f(T)$ 需要我们根据具体的数据, 给出 $focus$ 和 $T$ 的合理关系, 比如线性. 这些函数的具体算例我们将在问题五中给出.

### 5.2.2 动态合理底价估算模型

在通过多元线性回归得到静态底价估算模型的基础上, 我们对模型进行优化, 去进一步实现动态的分时段合理低价估算模型. 由于静态合理低价估算模型当中第一阶段对 focus 的求解要求各参数短期之内不变, 而在长期的情况下我们很难预测电视台的行为, 也就很难预测他们的变化趋势, 因此本文在这个模型中利用现有的收视率的数值对将来的收视率数值进行预测. 由于收视率的数值随时间的变化有明显的周期性, 故可以利用时间序列分析, 利用 ARMA 模型对时间序列 $focus$ 进行建模.

**定义 1 (自回归模型)** *如果 $focus$ 满足*

$$focus_t = \sum_{i=1}^{p} \beta_i focus_{t-i} + \varepsilon_t$$

*且$E(\varepsilon_t) = 0$, $Var(\varepsilon_t) > 0$, 则称时间序列 $focus$ 服从 $p$ 阶自回归模型, 记作 $AR(p)$.*

**定义 2 (移动平均模型)** *如果 $focus$ 满足*

$$focus_t = \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i} + \varepsilon_t$$

*则称时间序列 $focus$ 服从 $q$ 阶移动平均模型, 记作 $MA(q)$.*

**定义 3 (自回归滑动平均模型)** 如果 $focus$ 满足

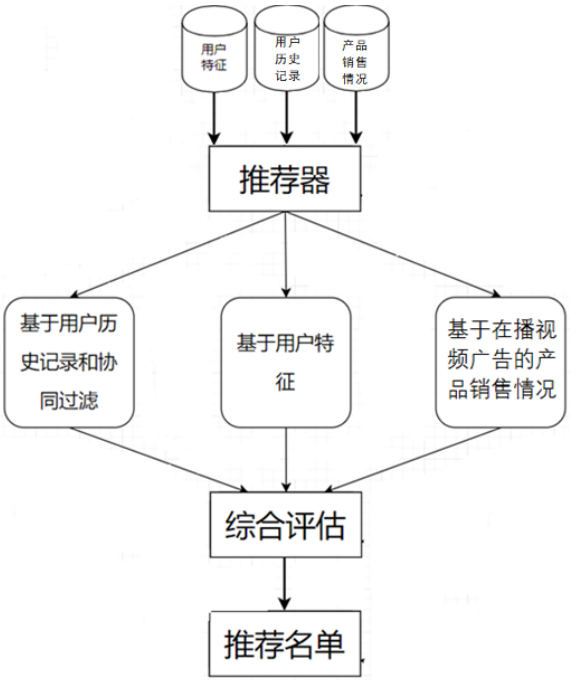
$$focus_t = \sum_{i=1}^{p} \beta_i focus_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}$$

则称时间序列 $focus$ 服从 $(p,q)$ 阶自回归滑动平均模型, 记作 $ARMA(p,q)$.

具体的算例将在问题五中给出.

## 5.3 问题三

模型总体上分为三个子部分, 分别是基于历史记录和协同过滤的推荐模型、基于用户特征的推荐模型以及基于在播视频广告的产品销售情况的推荐模型. 在输入端, 我们将用户的历史记录 $REC_1, REC_2, \cdots, REC_n$, $REC_i$ 记录用户看过的各类别广告次数、用户的特征 $A_1, A_2, \cdots, A_n$ 以及产品的销售情况 $s_1, s_2, \cdots, s_m$ 作为多标签输入, 将三种模型的输出归一化后, 根据该用户历史记录的数量计算三个子模型的相对置信度, 并以该置信度作为权重, 加权求和进行综合评估, 得到最终的匹配推送结果, 即每种广告对于该用户的推送概率 $P_i$. 模型的逻辑结构如下图所示:



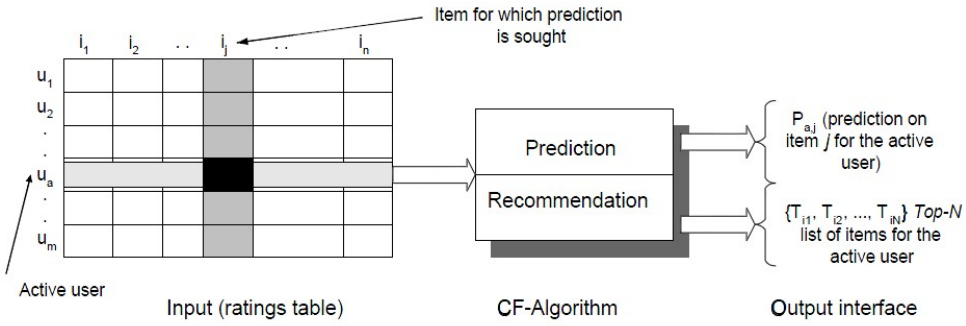

### 5.3.1 基于历史记录和协同过滤的推荐模型

在基于历史纪录的推荐模型中, 我们将模型分为两个部分, 分别是 Exploitation 和 Exploration. 其中, Exploitation 是基于用户的历史记录建立基于用户历史记录的推荐模型, 以较高的概率向用户推荐历史记录中较多的广告类型. 而 Exploration 模型是利用协

同过滤, 基于用户的历史记录以一定的概率向用户推荐历史纪录次数较低的广告. 综合以上两个部分, 可以实现在保证推荐的准确性的基础上, 尽量实现对用户喜爱的广告的挖掘.

我们设用户的历史记录向量为 $V$, $V_i$ 表示用户观看该种类型广告的次数.

在协同过滤部分, 我们输入的是用户历史记录向量, 输出的是各个广告类型的推荐概率向量. 协同过滤的总体流程如下图所示:




对于一个要进行广告推送的用户, 我们已知他之前对广告的历史行为, 我们采取寻找与他相似的若干用户, 将这些用户普遍喜好的商品广告推荐给该用户, 即根据和一个用户有共同喜好的人来给他推荐. 我们通过枚举计数将用户历史记录转化为向量 $x$, 其中 $x_i$ 表示 $x$ 用户对编号为 $i$ 的广告的喜好度.

具体在本问题中, 我们采用 Pearson 相关系数来表示相似度:

$$p(x,y) = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y}$$

其中 $s_x, s_y$ 代表样本标准差, 即$s_x = \frac{1}{n}\sum(x_i - \bar{x})^2$

在计算用户之间的相似度时, 是将一个用户对所有物品的偏好作为一个向量, 以求出相似度最高的 TOPK 相邻用户, 然后根据邻居的相似度权重以及它们对物品的偏好, 预测当前用户没有偏好的未涉及物品, 计算得到一个排序的物品列表进行推荐. 具体在本问题中, 计算相邻用户的方法是计算两个用户历史纪录的皮尔逊相关系数，根据皮尔逊相关系数的大小，对用户的近似程度进行排序，选取排名为前 K 个的用户。再利用如下所示的公式，计算用户历史纪录数为零的广告的 BSE 系数，并以 BSE 系数作为每个广告的推荐权重。其中 BSE 的计算公式以及匹配程度向量的计算公式如下所示：

$$BSE_i = p(x,y) \times (y_i - x_i)$$

$$P(V) = softmax(\alpha P_H(V) + \beta P_{CF}(V))$$

s.t. $\alpha + \beta = 1,\ \alpha >> \beta, P_H(V) = \frac{V}{\sum_{i=1}^{n} V_i}, P_{CF}(V) = \frac{\min(BSE_i)}{\sum_{i=1}^{n}\min(BSE_i)}$. $n$ 为协同过滤推荐的广告总数.

利用上述公式计算最终每类广告推荐的概率, 得到具有该种特征的用户对于所有类型广告的匹配程度向量 $P_1$. 至此我们可以采用协同过滤算法解决此问题.

### 5.3.2 基于用户特征的推荐模型

在问题一中，我们已经获得了基于用户的分类特征对视频广告进行推送的静态模型。利用用户的特征向量以及某一广告的特征向量，在遍历所有的广告类型后，我们可以得到具有该种特征的用户对于所有类型广告的匹配度 $L_i$。将 $L_i$ 归一化后，可以得到匹配程度向量 $P_2$。具体公式为：

$$P_2(x_i) = \frac{L_i}{\sum_{i=1}^{n} L_i}$$

### 5.3.3 基于在播视频广告的产品销售情况的推荐模型

由于较少的历史记录并不能有效说明用户对广告的喜爱程度, 所以当用户的历史记录较少时, 我们采用基于产品销售情况的推荐模型. 结合在播广告的产品销售情况, 我们采用当前最热推荐模型, 从极大似然估计的角度进行推荐. 我们将销售量高的广告以较高的概率推荐给用户. 具体在本问题中, 匹配程度的计算采用了 softmax 函数, 其公式为:

$$P(x_i) = softmax(X) = \frac{\exp X^T w_i}{\sum_{k=1}^{K} \exp x^T w_k}$$

由此, 我们可以得到具有该种特征的用户对于所有类型广告的匹配程度向量 $P_3$.

### 5.3.4 基于相对置信度的子模型加权融合

用户的历史记录数量越多, 越能体现用户的真实爱好, 进而使得基于历史记录和协同过滤的推荐模型准确率越高. 两者之间存在一定的映射关系. 为验证这一关系, 我们采用多项式进行回归

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

并得到了如下的结果:

| | |
|---|---|
| $\beta_0 = 0.1901$ | $(0.1868, 0.1934)$ |
| $\beta_1 = 0.00379$ | $(0.003732, 0.003849)$ |
| $\beta_2 = -1.056 \times 10^{-05}$ | $(-1.086 \times 10^{-05}, -1.026 \times 10^{-05})$ |
| $\beta_3 = 1.286 \times 10^{-08}$ | $(1.229 \times 10^{-08}, 1.342 \times 10^{-08})$ |
| $\beta_4 = -5.748 \times 10^{-12}$ | $(-6.098 \times 10^{-12}, -5.398 \times 10^{-12})$ |

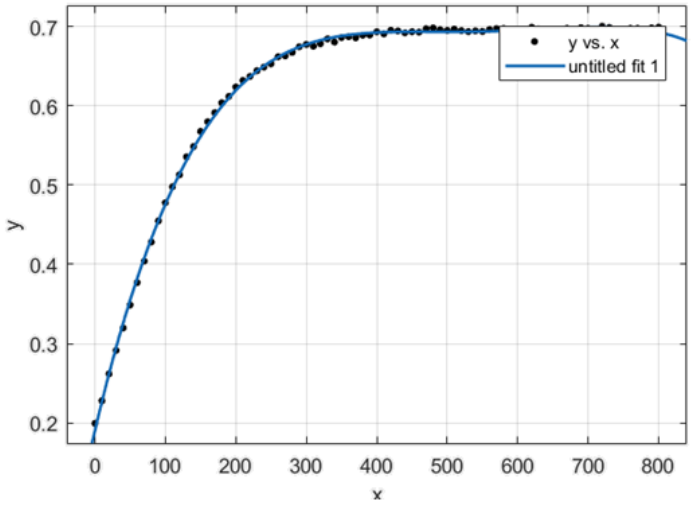



**图 3　协同过滤模型多项式拟合结果**

右侧为 95% 的置信区间, 从置信区间可以看出各项系数显著.

由此, 我们可以借助这个映射关系, 将用户据历史记录数量映射到协同过滤模型结果的准确率上, 并以该模型的准确率作为模型融合过程中该子模型的权重.

在子模型融合的过程中, 我们把三个模型的准确率作为各个模型的绝对置信度 $Ca$. 不难发现, 其中, 对于同一样本总体而言, 基于在播视频广告的产品销售情况的模型和基于用户特征模型的绝对置信度 $Ca$ 不随用户历史记录数量改变, 而基于用户历史和协同过滤的模型绝对置信度 $Ca$ 会随着用户的历史记录总数上升而上升.

我们将三个子模型的绝对置信度 $Ca$ 归一化为三者的相对置信度 $Cc$, 并把相对置信度 $Cc$ 作为子模型融合过程中的权重 $w$. 具体公式为 $C_ci = \dfrac{Ca_i}{\sum_{i=1}^{3} Ca_i}$

### 5.3.5 模型动态更新

当历史记录发生改变时, 基于在播视频广告的产品销售情况的推荐模型、基于用户历史记录和协同过滤的推荐模型及绝对置信度 $Ca_1$ 会实时发生改变, 进而完成对整个模型的动态修改. 由此, 我们确定了三个子模型的权重, 并根据三个子模型计算得出的匹配程度向量, 最终计算出推送给用户的广告类型概率向量 $P$, 并挑选 $P_i$ 最大的广告类型进行分户推送, 建立了" 广告 -用户" 分类匹配更新模型.

$$P = Cc_1 * P_1 + Cc_2 * P_2 + Cc_3 * P_3$$

## 5.4 问题四

由模型假设, 我们通过拍卖 [1], 使得电视台 ( 卖方) 与出价最高的销售商 ( 买方) 成交, 设成交价格为 $P$, 有 $N$ 个厂商参与本次拍卖, 记为 $A^1, A^2, \cdots, A^N$. 每个厂商心中的

底价为 $P^1, P^2, \cdots, P^N$, 则 $P = \max\limits_{i=1,2,\cdots,N} P^i$.

### 5.4.1 买方利益最大化的广告出价静态模型

本模型旨在给出在某一个时间点上, 什么样的条件下买方才会参与竞拍并且出多少的广告费以实现自身的利益最大化, 对此, 我们先给出买方的收入函数定义:

**定义 4 (买方的收入函数)** 设 $f(p, n, sell^i(p, N^*))$ 为买方的收入函数, $p$ 代表广告费用, $n$ 代表在该时刻该电视台投放广告的同一行业厂商数, $sell^i(p, N^*)$ 代表最近的生产周期厂商的销售量, 它与广告投入及当前的从业行业数有关.

由其定义及微观经济学知识 [3] 我们不难知道 $f$ 应具有下列性质:

- $f(0, n, sell^i(0))$ 代表不投入广告带来的收入, $f(0, n, sell^i(0))$ 及 $sell^i(0, N^*)$ 均应为正.
- $f$ 与投入的广告费 $p$ 正相关, 但是由于边际效益递减, 故每增加一单位的广告投入带来的收入增加越来越少. 即
$$\frac{\partial f}{\partial p} > 0 \qquad \frac{\partial^2 f}{\partial p^2} < 0$$
- 由于市场是有限的, $f$ 在 $\mathbb{R}^+ \times \mathbb{N} \times \mathbb{R}^+$ 上有上确界, $sell^i(p, N^*)$ 在 $\mathbb{R}^+$ 上有上确界.
- $f$ 与 $n$ 负相关, 因为 $n$ 增加, 则有更多的人抢占市场, 广告竞争更加的激烈, 收益变小, 即 $f$ 变小. 反之亦然. 在本模型的讨论中, $n$ 外生给定. [4]
- $\sum\limits_{i=1}^{N^*} sell^i(p^i) = D$, 即一段时间内需求 $D$ 不变, 也就是供给不变, 因此 $sell^i$ 可以被理解为一种市场份额. 厂商 $A^i$ 只能用上一阶段的广告费去估算自己的市场份额, 即销售量.

于是, 买方 $A^i$ 在投入 $p$ 元广告费的前提下, 其收益为:

$$F^i(p, n, sell^i(p, N^*)) = f^i(p, n, sell^i(p, N^*)) - p - VC^i(sell^i(p, N^*)) - FC^i$$

这里 $FC^i$ 为其不变成本, $VC^i(sell^i(p, N^*))$ 为其可变成本, 是销量的函数.

为使利润最大化, 我们有 $\dfrac{\partial F^i(p, n, sell^i(p, N^*))}{\partial p} = 0$, 即

$$f_1^i(p, n, sell^i(p, N^*)) + f_3^i(p, n, sell^i(p, N^*)) \frac{\mathrm{d}sell^i}{\mathrm{d}p} - 1 - \frac{\mathrm{d}VC^i(sell^i(p, N^*))}{\mathrm{d}sell^i} \frac{\mathrm{d}sell^i(p, N^*)}{\mathrm{d}p} = 0$$

**定理 1** $\varphi^i(p) = F^i(p, n, sell^i(p, N^*))$ 在 $[0, \infty)$ 上必有最大值.

**证明 1** 由模型的假设, 当 $p$ 很大的时候, 由于边际效益递减规律及市场的有限性, $f$ 有上确界, 故 $\exists p_0$, *s.t.*$\forall p > p_0$, $f^i(p, n, sell^i(p, N^*)) < p_0$, 此时 $\phi(p) < -VC^i(sell^i(p, N^*)) - FC^i < 0$. 由 $\varphi^i(p)$ 在 $[0, p_0]$ 的连续性, $\varphi^i(p)$ 在 $[0, p_0]$ 上一定有最大值.

另一方面, 由于 $\frac{\mathrm{d}\varphi^i(0)}{\mathrm{d}p} > 0$, 故 $\varphi^i$ 的极大值不可能在 $p = 0$ 时取得; 由上述分析, $\varphi^i$ 不可能在 $p = 0$ 时取得; 故 $\exists P^i$, *s.t.* $\forall p \in [0, p_0]$, $\varphi^i(P^i) > \varphi^i(p)$, $\forall p > p_0$, $\varphi^i(P^i) > 0 > \varphi^i(p)$. 故当 $p = P^i$ 时, $\varphi^i$ 取得极大值, *Q.E.D.*

于是, 为了使买方利益最大化, 买方 $A^i$ 在广告上的投入必为 $P^i$, 作为买方, 对于广告的效果是有所预期的. 这里, 我们假设为了投入 $P^i$ 的广告费, 而放弃的机会成本为 $\pi^i(p)$. 故若 $A^i$ 参与竞拍, 则必有

$$\varphi^i(P^i) > \varphi^i(0) + \pi^i(P^i)$$

该条件满足时, 买方 $A^i$ 才会参与竞拍.

### 5.4.2 买方选择卖方模型

这个模型中, 本文考虑对于买方 $A^i$, 该如何选择最好的卖方进行广告的投放.

**定义 5 (用户接受度)** 用户喜爱度的计算对于实现卖方对于自身最有决策的选择至关重要, 广告喜爱度体现了用户对于该广告的普遍喜爱程度. 在两个竞价方出价相同时, 如果某一方的广告的用户喜爱度较高则可以说明该广告更能起到提升收视率和销售量的效果, 对于电视台的长期收益是更为有益的.

对于每一个电视台 $T^j$, 买方 $A^i$ 的广告有一个用户接受度 $U$. 利用前文中提出的更新的分户匹配推送模型可以对用户喜爱度进行计算, 计算的方式如下:

1. 利用更新的分户匹配推送模型, 对用户池中所有的用户对所有类别的广告进行一次用户喜爱度 $L$ 的预测, 每个竞价周期只需要进行一次计算, 且该计算的结果可以在后续的分户推荐中起到至关重要的作用.
2. 得到了每一个用户对每一类别的用户喜爱度 $L_{ij}$ 后, 对用户喜爱度进行归一化处理.
3. 将归一化后的用户喜爱度对每一种类型的广告进行求和, 以此估计出某一种广告在当前用户中的用户接受度 $U$.

$$U = \sum_{i=1}^{n} \frac{L_{ij}}{\sum_{j=1}^{m} L_{ij}}$$

利用该种方式计算出的用户接受度可以在宏观上表示该类产品在电视台观众处的接受程度. 其可以由第三问的结论给出.

于是, 买方优先考虑选择用户接受度高的去竞拍.

### 5.4.3 卖方利益最大化的静态模型

作为卖方, 我们可以根据问题 5.2 的结果, 根据某节目的类型, 节目的播放时间, 欲出售的广告所在位置 ( 该节目播放前、中、后), 来确定一个拍卖底价 $P^0$. 卖方采用广义的第二密封竞价模型, 买方之间不知道彼此的出价金额, 只能知道自己在竞争中的排名是不是第一位, 一旦交易成功, 中标者支付价格的公式为

$$P = \frac{P_{-1}U_{-1}}{U_0}$$

这里设出价最高的买家为 $A^0$, 第二高的买家为 $A^{-1}$, 对应的用户接受度记为 $U_0, U_{-1}$.

**定理 2** 卖方实现了利益最大化.

**证明 2** 由于交易时买家之间并不知道彼此的报价, 所以可以达到出价的纳什均衡, 每个出价者有足够的激励去报出自己能承受的最高报价. 在每个买家报出自己能承受的最大报价的情况下, 卖方能够得到相对最大的收益. ”一价”会导致系统抖动, 造成不稳定. 但”二价”中, 由于买家的计费价格跟自己的出价没有直接关系, 没有动力通过修改自己的出价, 在赢得拍卖的同时获得更多的利益, 因此避免了出价的抖动.

采用密封的形式可以有效地避免竞价方得知其他人的报价后, 选择对自己更有利但对电视台的收益不利的选择. 可以实现对电视台收益的最大化.

### 5.4.4 动态竞价交易模型

我们现在基于 5.4.1 及 5.4.2 提出的静态模型建立动态模型. 我们考虑从第 $t-1$ 时刻到第 $t$ 时刻的状态转移.

在 $t-1$ 时刻, 我们知道卖方的底价 $P_{t-1}^0$, 某一行业在该电视台有 $n_{t-1}$ 个广告在播, 该行业的厂商总数为 $N_{t-1}^*$, 参与竞标的厂商总数为 $N_{t-1}$, 厂商 $A^i$ 在最近一个生产周期的广告投入 $P_{t-1}^i$.

则在 $t$ 时刻, 厂商预测在 $t$ 时刻会有 $n_e$ 个厂商的广告被播出, 将这种预测作为真实值的猜测. 根据模型 5.4.1, 可以计算出厂商 $A^i$ 愿意投入的广告费 $P_t^i$; 根据模型 5.4.2, 卖方的底价为 $P_t^0$; 设此时该行业的厂商为 $N_t^*$; 此时参与竞标的厂商数为

$$N_t = \sum_{i=1}^{N_t^*} I[F^i(P_t^i, n_{t-1}, sell^i(P_t^i, N_{t-1}^*)) > F^i(0, n_{t-1}, sell^i(P_t^i, N_{t-1}^*)) + \pi^i(P_t^i)]$$

这里 $I$ 为示性函数, 里面的表达式为真取值为 1, 否则为 0.

根据博弈论中不完全信息动态博弈 [1] 的结果, 其他的厂商知道 $A_i$ 会用上述方法计算自己的广告投入, 于是其他的厂商在算 $A^i$ 的市场份额时, 将不会用 $P_{t-1}^i$, 而会用通过上述算法得到的 $\hat{P}^i$. 另一方面厂商 $A^i$ 猜测其他厂商会这么做, 于是 $A^i$ 也会调整自己的价格, 这样的过程一直下去, 所有的厂商必然会收敛到一个均衡价格 [1], 我们这里继续沿用上面的符号 $P_t^i$.

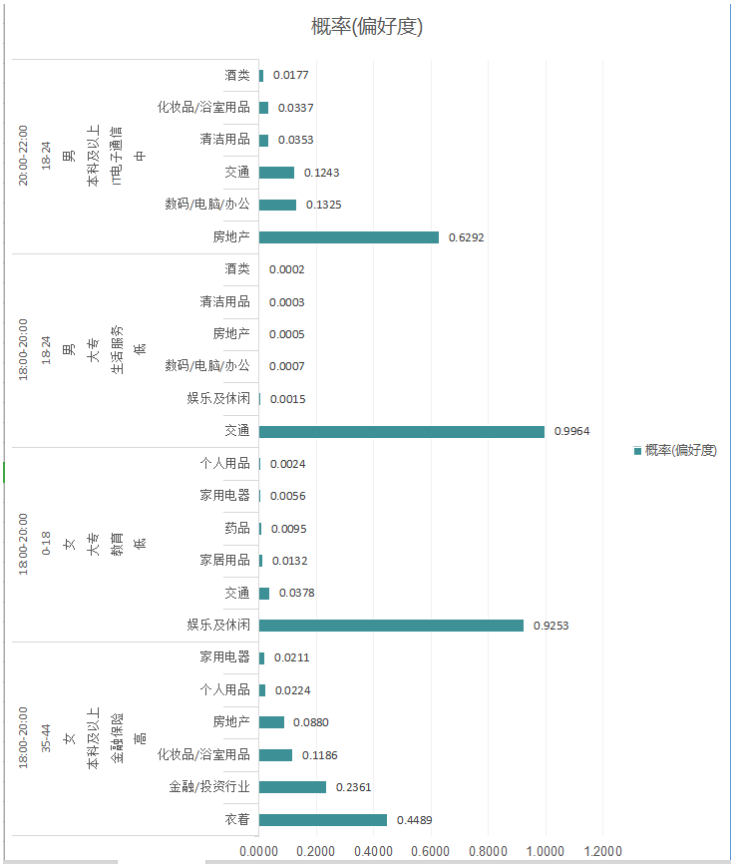
同时, 对于广告商来说, 同样采用模型 5.4.3 的第二密封竞价拍卖模型. 在所有的竞价均完成后, 我们可以得到某一行业在该电视台有 $n_t$ 个广告在播, 这是真实值. 这样我们就完成了两个时间层的状态转移. 得到广告的编排. 需要强调的是, 用户的体验在这里起到了非常关键的作用, 尤其是影响到用户接受度 $U$ 这一参数, 使得广告商必须要去考虑自己广告的接受度, 电视台需要考虑这些广告带来的效益. 具体的博弈过程算例在问题五中给出.

## 5.5 问题五

我们将会结合数据给出算例:

### 5.5.1 第一问算例

我们结合模型, 进行了 2000 组测试[1], 给出以下算例. 我们选取其中 4 个展示.




输入为一个 28 维的特征向量, 输出是所有广告中经过 sigmoid 函数后的匹配概率最大的六个广告类型, 即匹配概率代表用户对此广告类型的偏好度. 偏好度越高, 广告主应对其选择推送.

### 5.5.2 问题二算例

对于问题二的实证分析, 我们基于多元线性回归 [5], 以 CCTV1( 综合类频道的代表), CCTV12( 法制频道, 专题类代表), 湖南卫视 ( 地方卫视的代表) 为例来讨论.

**1. CCTV1 合理底价估算模型** 对于 CCTV1, 我们将其节目类型分为法制专题, 电视剧, 综艺, 新闻四种类型, 对应虚拟变量 $D_{11}$, $D_{12}$, 将广告播放时段分为节目 ( 播出) 前, 节目中插播, 节目 ( 结束) 后, 对应虚拟变量 $D_{21}$, $D_{22}$, 对应关系如下:

[1]其余见附录.

| 节目类型 | $D_{11}$ | $D_{12}$ | $D_{13}$ |
|---|---|---|---|
| 专题 | 1 | 0 | 0 |
| 电视剧 | 0 | 1 | 0 |
| 综艺 | 0 | 0 | 0 |
| 新闻 | 0 | 0 | 0 |

| 插播时段 | $D_{21}$ | $D_{22}$ |
|---|---|---|
| 节目前 | 1 | 0 |
| 节目中 | 0 | 1 |
| 节目后 | 0 | 0 |

利用我们已知的数据, 我们希望预测出合理的底价. 首先我们希望能根据节目的类型以及其播放时段、节目播出时间相对值 $t$, 来预测其直播关注度 $focus$ 建立如下的多元线性回归模型:

$$\begin{aligned} FOCUS =&\beta_1 D_{12}|T-0.5| + \beta_2 D_{12}|T-0.8| \\ &+ \beta_3|T-0.5| + \beta_4 D_{13}T + \beta_0 \end{aligned}$$

这里, 考虑到中午 12 点及晚上 19 点是两个收视高峰, 为提高模型的准确性, 我们分别用 $t-0.5$, $|t-0.8|$ 来衡量一天中播出时间与中午 12 点及晚上 19 点之间的差距, 差距越大说明在同一天内, 播出时间分别离 12 点、19 点越远. 我们先对上述方程进行进行回归, 得到的回归方程为:

$$\begin{aligned} \widehat{focus} =&0.04662 D_{12}|T-0.5| - 0.06139 D_{12}|T-0.8| + 0.07177 D_{13}T \\ &- 0.058499|T-0.5| + 0.02583 \end{aligned}$$

然后我们建立市占率 $rate$ 以及直播关注度 $focus$, 虚拟变量 $D_{11}$, $D_{12}$, $D_{21}$, $D_{22}$ 的关系, 剔除不显著变量以后, 得到的回归方程为:

$$\widehat{rate} = 0.3539 focus + 3.4877 D_{12} focus - 0.01007(D_{11}+D_{12}) + 0.02759 D_{13} + 0.06727$$

最后, 我们通过已知数据中的市占率 $rate$ 以及广告时长 $adv$, 来推测价格的平均值, 我们在 Eviews 中回归, 结果如下:

$$\begin{aligned} \widehat{\frac{P}{adv}} =& -528822.8227 D_{22} focus + 490186.7586 focus \\ &- 3746.5851 D_{11} - 4376.9636 D_{12} + 5829.23358572 D_{22} + 6885.64196555 \end{aligned}$$

将上述方程合并消去 $rate$, $focus$, 最后得到的方程如下:

$$\begin{aligned} \widehat{\frac{P}{adv}} =&5829.2336 D_{22} - 4376.9636 D_{12} - 3746.585 D_{11} - 28675.489|T-0.5| \\ &- 30093.279 D_{12}|T-0.8| - 528822.8227 D_{22}(0.07177 D_{13}T - 0.06139 D_{12}|T-0.8| \\ &- 0.058499|T-0.5| + 0.04662 D_{12}|T-0.5| + 0.02583) + 35181.485 D_{13}T \\ &+ 22852.9634 D_{12}|T-0.5| + 19547.0048 \end{aligned}$$

**2. CCTV12 合理底价估算模型** 对于 CCTV12, 我们将其节目类型分为法制专题, 电视剧, 综艺三种类型, 对应虚拟变量 $D_{11}$, $D_{12}$, 将广告播放时段分为节目 ( 播出) 前, 节目中插播, 节目 ( 结束) 后, 对应虚拟变量 $D_{21}$, $D_{22}$, 对应关系如下:

| 节目类型 | $D_{11}$ | $D_{12}$ |
|---|---|---|
| 法制专题 | 1 | 0 |
| 电视剧 | 0 | 1 |
| 综艺 | 0 | 0 |

| 插播时段 | $D_{21}$ | $D_{22}$ |
|---|---|---|
| 节目前 | 1 | 0 |
| 节目中 | 0 | 1 |
| 节目后 | 0 | 0 |

利用我们已知的数据, 我们希望预测出合理的底价. 首先我们希望能根据节目的类型以及其播放时段、节目播出时间相对值 $t$, 来预测其直播关注度 $focus$ 建立如下的多元线性回归模型:

$$focus = \beta_0 + \beta_{11}D_{11} + \beta_{12}D_{12} + \beta_{21}D_{21} + \beta_{22}D_{22} + \beta_3|t - 0.8|$$

这里, 考虑到中午 12 点及晚上 18 点是两个收视高峰, 为提高模型的准确性, 我们用 $|t - 0.8|$ 来衡量一天中播出时间与晚上 19 点之间的差距, 差距越大播出时间离 19 点越远. 我们先对上述方程进行进行回归, 得到的回归方程为:

$$\begin{aligned}
\widehat{focus} = \quad & -0.00269|T - 0.8| + \quad 0.0001896D_{11} + \quad 0.0003705D_{12} \\
& (0.0000) \qquad\qquad (0.4336) \qquad\qquad (0.1349) \\
& -0.0002305D_{21} \qquad -6.8498 \times 10^{-5}D_{22} + \quad 0.001775 \\
& (0.3551) \qquad\qquad (0.7823) \qquad\qquad (0.0000) \\
& R = 0.647745 \qquad F = 72.81846 \qquad P = 0.000000
\end{aligned}$$

我们看到, $D_{11}$, $D_{12}$, $D_{21}$, $D_{22}$ 的系数是不显著的, 我们将其剔除, 得到以下回归方程

$$\begin{aligned}
\widehat{focus} = \quad & -0.0026588|T - 0.8| + \quad 0.001868 \\
& (0.0000) \qquad\qquad (0.0000) \\
& R = 0.627768 \qquad F = 340.6727 \qquad P = 0.000000
\end{aligned}$$

然后我们建立市占率 $rate$ 以及直播关注度 $focus$, 虚拟变量 $D_{11}$, $D_{12}$, $D_{21}$, $D_{22}$ 的关系, 剔除不显著变量以后, 得到的回归方程为:

$$\begin{aligned}
\widehat{rate} = \quad & 2.9231 * focus + \quad 0.005970D_{12} \quad 0.009411 \\
& (0.0000) \qquad\qquad (0.0000) \qquad (0.0000) \\
& R = 0.508964 \qquad F = 104.1692 \qquad P = 0.000000
\end{aligned}$$

最后, 我们通过已知数据中的市占率 $rate$ 以及广告时长 $adv$, 来推测价格的平均值, 我们在 Eviews 中回归, 结果如下:

$$\begin{array}{llll} \widehat{\frac{P}{adv}} = & 1935.91 * adv & 2481762 * rate- & 29602.6146441 D_{12} \quad 15643.29 \\ & (0.0000) & (0.0001) & (0.0000) \qquad\qquad (0.0934) \\ & R = 0.254679 & F = 22.78025 & P = 0.000000 \end{array}$$

将上述方程合并消去 $rate$, $focus$, 最后得到的方程如下:

$$\widehat{P} = 1935.9104 adv - 14786.7542 D_{12} - 19288.5831|t - 0.8| + 21266.232 \tag{1}$$

可见, 底价和广告时长, 是否为电视剧, 播放时间和 19 点之间的距离有关系.

**3. 湖南卫视合理底价估算模型** 对于湖南卫视, 我们将其节目类型分为电视剧, 综艺, 新闻三种类型, 对应虚拟变量 $D_{11}$, $D_{12}$, 将广告播放时段同上分为节目 ( 播出) 前, 节目中插播, 节目 ( 结束) 后, 对应虚拟变量 $D_{21}$, $D_{22}$, 对应关系如下:

| 节目类型 | $D_{11}$ | $D_{12}$ |
|---|---|---|
| 电视剧 | 1 | 0 |
| 综艺 | 0 | 1 |
| 新闻 | 0 | 0 |

| 插播时段 | $D_{21}$ | $D_{22}$ |
|---|---|---|
| 节目前 | 1 | 0 |
| 节目中 | 0 | 1 |
| 节目后 | 0 | 0 |

利用我们已知的数据, 我们希望预测出合理的底价. 首先我们希望能根据节目的类型以及其播放时段、节目播出时间相对值 $t$, 来预测其直播关注度 $focus$. 按照节目类型通过散点图观察 $rate$ 和 $t$ 之间的关系, 我们可以建立如下的多元线性回归模型:

$$\begin{aligned} focus =& \beta_0 + \beta_1 D_{11} T + \beta_2 |T - 0.3| + \beta_3 |T - 0.5|^2 + \beta_{41} D_{11} |T - 0.5| + \beta_{42} D_{12} |T - 0.5|^3 + \\ & \beta_5 |T - 0.9| + \beta_6 D_{11} |T - 0.9| + \beta_7 D_{12} |T - 0.85|^{0.5} \end{aligned}$$

这里, 考虑到中午 12 点及晚上 18 点是两个收视高峰, 为提高模型的准确性, 我们用 $|t - 0.8|$, 来衡量一天中播出时间与晚上 19 点之间的差距, 差距越大播出时间离 19 点越远. 我们先对上述方程进行进行回归, 剔除不显著的变量, 得到以下回归方程:

$$\begin{aligned} \widehat{focus} =& 0.1056 D_{11} T - 0.3761 |T - 0.3| + 0.5766 |T - 0.5|^2 - 0.2777 D_{11} |T - 0.5| \\ & - 0.2785 D_{12} |T - 0.5|^3 - 0.2191 |T - 0.9| - 0.05596 D_{11} |T - 0.9| + \\ & 0.01575 D_{12} |T - 0.85|^{0.5} + 0.1661 \\ & R = 0.627768 \qquad F = 340.6727 \qquad P = 0.000000 \end{aligned}$$

然后我们建立市占率 $rate$ 以及直播关注度 $focus$, 虚拟变量 $D_{11}$, $D_{12}$, $D_{21}$, $D_{22}$ 的关系, 剔除不显著变量以后, 得到的回归方程为:

$$\begin{aligned}\widehat{rate} =& -6.811focus + 7.4645D_{11}focus + 9.467D_{12}focus \\ & - 0.03183D_{11} - 0.03177D_{12} + 0.06875 \\ & R = 0.432745 \qquad F = 0.432745 \qquad P = 0.000017\end{aligned}$$

最后, 我们通过已知数据中的市占率 $rate$ 以及广告时长 $adv$, 来推测价格的平均值, 我们在 Eviews 中回归, 结果如下:

$$\begin{aligned}\widehat{P} =& 3377.6842adv + 701623.8718 * rate + \\ & 35282.3786D_{12} + 25743.5631D_{21} - 40738.4427 \\ & R = 0.441840 \qquad F = 10.29082 \qquad P = 0.000003\end{aligned}$$

将上述方程合并消去 $rate$, $focus$, 最后得到的方程如下:

$$\begin{aligned}\widehat{P} =& 3377.6842adv - 22333.3937D_{11} + 12991.4813D_{12} + 25743.5631D_{21} + 1797414.2932|t-0.3| \\ & + 1046930.8284|t-0.9| - 504898.6989D_{11}t + 1327354.22D_{11}|t-0.5| \\ & + 267417.5809D_{11}|t-0.9| - 5237295.04563D_{11}(0.3761|t-0.3| + 0.219|t-0.9| - 0.1056D_{11}t \\ & + 0.2777D_{11}|t-0.5| + 0.05596D_{11}|t-0.9| + 0.2785D_{12}|t-0.5|^3 - 0.01575D_{12}|t-0.85|^{0.5} \\ & - 0.57665|t-0.5|^2 - 0.166) - 6642297.022D_{12}(0.3761|t-0.3| + 0.219|t-0.9| - 0.1056D_{11}t \\ & + 0.2777D_{11}|t-0.5| + 0.05596D_{11}|t-0.9| + 0.2785D_{12}|t-0.5|^3 - 0.01575D_{12}|t-0.85|^{0.5} \\ & - 0.57665|t-0.5|^2 - 0.166) + 1330920.1989D_{12}|t-0.5|^3 - 75266.7893D_{12}|t-0.85|^{0.5} \\ & - 2755823.893|t-0.5|^2 - 786003.684\end{aligned}$$

**4.CCTV1 时间序列模型** 本文爬取到 CCTV1 从 2019.03.01 到 2019.04.25 日每 3 个小时的收视率数据, 建立 ARMA 自回归滑动平均模型.

先用 ADF 检验验证 $focus$ 的平稳性, ADF 检验是通过下述三个模型完成的:

$$\Delta X_t = \delta X_{t-1} + \sum_{i=1}^{m} \Delta X_{t-i} + \varepsilon_t$$

$$\Delta X_t = \delta X_{t-1} + \sum_{i=1}^{m} \Delta X_{t-i} + \alpha + \varepsilon_t$$

$$\Delta X_t = \delta X_{t-1} + \sum_{i=1}^{m} \Delta X_{t-i} + \alpha + \beta T + \varepsilon_t$$

这里 $m$ 为滞后阶数, 本文取定为 1. 在 EViews 软件中完成该检验, 结果如下图所示:

Null Hypothesis: FOCUS has a unit root
Exogenous: None
Lag Length: 1 (Fixed)

| | | t-Statistic | Prob.* |
|---|---|---|---|
| Augmented Dickey-Fuller test statistic | | -8.104187 | 0.0000 |
| Test critical values: | 1% level | -2.570157 | |
| | 5% level | -1.941535 | |
| | 10% level | -1.616223 | |

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(FOCUS)
Method: Least Squares
Date: 06/04/19 Time: 09:15
Sample (adjusted): 3 448
Included observations: 446 after adjustments

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS(-1) | -0.349765 | 0.043159 | -8.104187 | 0.0000 |
| D(FOCUS(-1)) | -0.175796 | 0.046993 | -3.740867 | 0.0002 |

| | | | |
|---|---|---|---|
| R-squared | 0.236426 | Mean dependent var | 4.13E-05 |
| Adjusted R-squared | 0.234706 | S.D. dependent var | 0.661527 |
| S.E. of regression | 0.578711 | Akaike info criterion | 1.748448 |
| Sum squared resid | 148.6986 | Schwarz criterion | 1.766835 |
| Log likelihood | -387.9039 | Hannan-Quinn criter. | 1.755698 |
| Durbin-Watson stat | 2.166923 | | |

(a) 模型 1

Null Hypothesis: FOCUS has a unit root
Exogenous: Constant
Lag Length: 1 (Fixed)

| | | t-Statistic | Prob.* |
|---|---|---|---|
| Augmented Dickey-Fuller test statistic | | -23.63683 | 0.0000 |
| Test critical values: | 1% level | -3.444790 | |
| | 5% level | -2.867801 | |
| | 10% level | -2.570168 | |

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(FOCUS)
Method: Least Squares
Date: 06/04/19 Time: 09:17
Sample (adjusted): 3 448
Included observations: 446 after adjustments

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS(-1) | -1.506922 | 0.063753 | -23.63683 | 0.0000 |
| D(FOCUS(-1)) | 0.403109 | 0.043631 | 9.238957 | 0.0000 |
| C | 0.839030 | 0.040479 | 20.72733 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.612360 | Mean dependent var | 4.13E-05 |
| Adjusted R-squared | 0.610610 | S.D. dependent var | 0.661527 |
| S.E. of regression | 0.412801 | Akaike info criterion | 1.074999 |
| Sum squared resid | 75.48909 | Schwarz criterion | 1.102580 |
| Log likelihood | -236.7248 | Hannan-Quinn criter. | 1.085874 |
| F-statistic | 349.9062 | Durbin-Watson stat | 2.015539 |
| Prob(F-statistic) | 0.000000 | | |

(b) 模型 2

Null Hypothesis: FOCUS has a unit root
Exogenous: Constant, Linear Trend
Lag Length: 1 (Fixed)

| | | t-Statistic | Prob.* |
|---|---|---|---|
| Augmented Dickey-Fuller test statistic | | -23.75510 | 0.0000 |
| Test critical values: | 1% level | -3.978767 | |
| | 5% level | -3.419930 | |
| | 10% level | -3.132602 | |

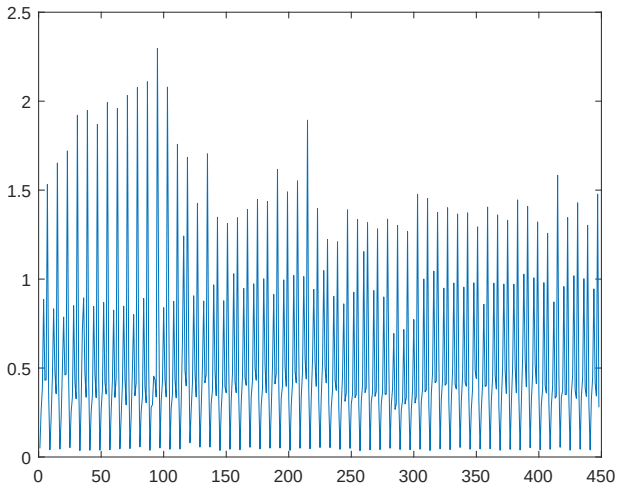
*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(FOCUS)
Method: Least Squares
Date: 06/04/19 Time: 09:17
Sample (adjusted): 3 448
Included observations: 446 after adjustments

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS(-1) | -1.514683 | 0.063762 | -23.75510 | 0.0000 |
| D(FOCUS(-1)) | 0.407128 | 0.043592 | 9.339443 | 0.0000 |
| C | 0.902779 | 0.054479 | 16.57112 | 0.0000 |
| @TREND("1") | -0.000265 | 0.000152 | -1.743509 | 0.0819 |

| | | | |
|---|---|---|---|
| R-squared | 0.615008 | Mean dependent var | 4.13E-05 |
| Adjusted R-squared | 0.612395 | S.D. dependent var | 0.661527 |
| S.E. of regression | 0.411853 | Akaike info criterion | 1.072630 |
| Sum squared resid | 74.97347 | Schwarz criterion | 1.109404 |
| Log likelihood | -235.1964 | Hannan-Quinn criter. | 1.087129 |
| F-statistic | 235.3582 | Durbin-Watson stat | 2.022052 |
| Prob(F-statistic) | 0.000000 | | |

(c) 模型 3




我们看到三个模型都显著的没有单位根, 故我们认为 $focus$ 是平稳的. 对于平稳的序列, 是可以进行 ARMA 建模的. 下面确定 $(p, q)$ 的阶数.

我们发现数据具有明显的 8 个时段的周期性, 且数据整体有波动趋势, 故对数据做下列差分运算: $f_t = focus_{t+8} - focus_t$, 用选取的 p,q 的各种阶数进行试算, 用 AIC 和 BIC 准则进行定阶.

| p | q | AIC | BIC | p | q | AIC | BIC |
|---|---|---|---|---|---|---|---|
| 0 | 1 | -754.7434 | -746.5744 | 2 | 0 | -657.1869 | -644.9334 |
| 0 | 2 | -771.5606 | -759.3072 | 2 | 1 | -769.9108 | -753.5728 |
| 0 | 3 | -770.7636 | -754.4256 | 2 | 2 | -768.5359 | -748.1134 |
| 1 | 0 | -627.8779 | -619.7089 | 2 | 3 | -788.1413 | -763.6343 |
| 1 | 1 | -771.9092 | -759.6557 | 3 | 0 | -684.6975 | -668.3595 |
| 1 | 2 | -770.1643 | -753.8263 | 3 | 1 | -768.8152 | -748.3927 |
| 1 | 3 | -768.8044 | -748.3819 | 3 | 2 | -772.7188 | -748.2118 |
| 3 | 3 | -783.4131 | -754.8216 | | | | |

从表中我们可以看出, 取定 $(p, q) = (2, 3)$, MATLAB 的报告为

| | Value | StandardError | TStatistic | PValue |
|---|---|---|---|---|
| Constant | 0 | 0 | NaN | NaN |
| AR1 | -1.1627 | 0.049868 | -23.316 | 3.0476e-120 |
| AR2 | -0.68833 | 0.05331 | -12.912 | 3.864e-38 |
| MA1 | 0.35762 | 0.030656 | 11.666 | 1.9073e-31 |
| MA2 | -0.45145 | 0.032232 | -14.007 | 1.4217e-44 |
| MA3 | -0.90617 | 0.03324 | -27.261 | 1.2239e-163 |
| Variance | 0.0094615 | 0.00040902 | 23.132 | 2.1896e-118 |

此时的 ARMA 模型为

$$f_t = -1.1627 f_{t-1} - 0.6833 f_{t-2} + \varepsilon_t + 0.35762\varepsilon_{t-1} - 0.45145\varepsilon_{t-2} - 0.90617\varepsilon_{t-3}$$

### 5.5.3 第三问算例

我们在数据集中选取了 10 个用户进行模型的算例演示, 为简便的演示推荐模型, 我们假设广告的类型只有 10 种, 在数据集中只选取 10 种广告类型进行演示. 这 10 个人的历史纪录和特征如下表所示:

| 用户 id \广告种类 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 2 | 4 | 5 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 6 | 2 | 0 | 0 | 1 | 0 | 7 | 5 | 7 | 0 |
| 3 | 9 | 6 | 3 | 0 | 2 | 0 | 0 | 1 | 4 | 7 |
| 4 | 5 | 2 | 1 | 3 | 2 | 0 | 5 | 6 | 9 | 1 |
| 5 | 7 | 5 | 6 | 1 | 0 | 5 | 3 | 8 | 9 | 4 |
| 6 | 10 | 2 | 4 | 5 | 0 | 0 | 0 | 0 | 8 | 0 |
| 7 | 9 | 8 | 3 | 6 | 0 | 0 | 5 | 0 | 1 | 0 |
| 8 | 8 | 7 | 0 | 0 | 1 | 2 | 3 | 6 | 0 | 5 |
| 9 | 5 | 0 | 2 | 3 | 3 | 1 | 0 | 0 | 5 | 6 |
| 10 | 5 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |

假设当前十种广告的产品销售情况为下表所示（归一化后）:

| 广告种类 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 销售情况 | 0.0807 | 0.0904 | 0.0515 | 0.1168 | 0.0090 | 0.1161 | 0.1613 | 0.1269 | 0.0743 | 0.0156 |

假设我们向用户 1 进行广告推送, 则我们只需关注用户 1 的特征, 用户 1 的特征用数值量化后如下表所示:

| 年龄 | 性别 | 地域 | 婚恋状态 | 学历 | 消费水平 | 工作状态 |
|---|---|---|---|---|---|---|
| 5 | 0 | 15 | 3 | 5 | 4 | 3 |

于是, 基于该用户的特征, 利用第一问中我们通过 DeepFm 获得的用户特征与广告特征之间的分类匹配推送模型, 我们可以获得该用户对于这十种广告的匹配度 $Like$（归一化后）分别为:

| 广告种类 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 匹配度 Like | 0.0098 | 0.0373 | 0.399 | 0.2157 | 0.0545 | 0.0237 | 0.0015 | 0.0495 | 0.016 | 0.0118 |

由此, 基于用户特征与广告特征的分类匹配推送静态模型, 我们得到了匹配程度向量. $P_2 = (0.009813, 0.037313, 0.399064, 0.215716, 0.054545, 0.023783, 0.001544 + 0.049562 + 0.0162 + 0.011811)$

基于当前在播广告的归一化后的产品销售情况, 假设 softmax 函数对于每一个广告的权重都为 1, 则进行计算后可以得出匹配程度向量

$P_3 = (0.099539, 0.100507, 0.096671, 0.103196, 0.092654, 0.103128, 0.107896, 0.104243, 0.098900, 0.093266)$

在该算例中, 我们设基于用户特征的推荐模型的绝对准确率 $Ca_1 = 0.8$, 基于在播视屏广告的产品销售情况的推荐模型准确率 $Ca_2 = 0.2$, 由于用户的历史记录总数

基于用户 1 的历史记录，我们利用 Exploration 和 Exploitation 进行推荐。在该算例中，我们设 Exploration 的推荐权重为 0.2，Exploitation 的权重为 0.8。其中 Exploitation 的计算结果为 PH=(0.409091,0.0909090.181818,0.227273,0.045455,0,0.045455,0,0,0) 而利用 Exploration 进行协同过滤，我们可以得到用户 1 与其他用户之间的皮尔逊相关系数如下表。

| 用户id | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 皮尔逊相关系数 | 0.0072 | 0.0546 | 0.0004 | -0.0011 | 0.0796 | 0.0904 | 0.0320 | 0.0480 | 0.1756 |

在该算例中，我们设协同过滤模型中 Top-K 的近似用户中的 K 值取 1，即只选择与用户 1 最相似的用户 10，经过计算后，我们可以得到 Pcf=(0,0,0,0,0,0,0,0,0.3333,0.6667) 综合 Exploration 和 Exploitation，我们得到最终的匹配向量 P1=(0.12492,0.09684, 0.10456 0.108013 0.12955 0.09006 0.12955 0.09006 0.09626 0.1029 )

### 5.5.4 第四问算例

我们讨论市场上只有一家广告供给方 ( 卖方), 两家企业美的, 格力. 初始状态下, 他们都选择不打广告, 即 $p_0^1 = p_0^2 = 0$, 此时他们的市场份额相同, 即销售量相同. 由于格力做空调的技术更加成熟, 因此同样的产品, 其成本相对于美的更低. 设市场的总需求 $D = 100000$, 空调价格 $P = 1000$, 格力公司成本 $C^1 = 100$, 美的公司成本 $C^2 = 300$, 他们的不变成本 $FC = 5000000$, 则不打广告时的销量为 $Q = 50000$. 设格力的出价为 $p^1$, 美的的出价为 $p^2$, 则格力公司的销量为 $sell^1 = D\dfrac{p^1 + Q}{p^1 + p^2 + 2Q}$, 美的公司的销量为 $sell^2 = D\dfrac{p^2 + Q}{p^1 + p^2 + 2Q}$. 格力公司的利润为 $profit^1 = 900sell^1 - FC - 1.03p^1$, 美的公司的利润 $profit^2 = 700sell^2 - FC - 1.03p^2$. 这里 $1.03p^i$ 是若将 $p^i$ 元存入银行, 获得的本利和, 作为机会成本.

我们这里描述在一个时间点上的博弈过程:

**Step 1** 先求出两家公司第一轮博弈的出价 $p_1^1, p_1^2$. 格力公司预测美的公司不会加大广告投入, 即已知 $p_1^2 = p_0^2 = 0$ 的前提下最大化自身的利润, 解得

$$p_1^1 = \sqrt{\frac{900 \times 10^5 \times (50000 + p_1^2)}{1.03}} - p_1^2 - 100000 = 1990199.04$$

**Step 2** 同理对于美的公司, 美的预测格力的出价为 $p_1^1$, 用同样的方法去求出 $p_1^2 = 9684597.172$

**Step 3** 对于格力, 重新预测美的的出价 $p_1^2$, 更新自己的出价 $p_2^1 = 19380558$

依次类推, 最后的出价一定收敛 [1], 且收敛于 $(p^{1*}, p^{2*})$, 其中 $p^{1*}, p^{2*}$ 必为方程组

$$\begin{cases} \dfrac{\partial profit^1(p^{1*}, p^{2*})}{\partial p^1} = 0 \\ \dfrac{\partial profit^2(p^{1*}, p^{2*})}{\partial p^2} = 0 \end{cases}$$

的解, 这里的收敛过程为:

| 迭代次数 $i$ | $p_i^1$ | $p_i^2$ |
|---|---|---|
| 1 | 1990199.042875 | 9684957.172000 |
| 2 | 19380558.865072 | 16858447.973511 |
| 3 | 21478997.331412 | 16671960.054904 |
| 4 | 21452928.966776 | 16674863.338537 |
| 5 | 21453343.861087 | 16674817.240893 |
| 6 | 21453337.275745 | 16674817.972598 |
| 7 | 21453337.380274 | 16674817.960984 |
| 8 | 21453337.378615 | 16674817.961168 |
| 9 | 21453337.378641 | 16674817.961165 |
| 10 | 21453337.378641 | 16674817.961165 |

此时双方的利润分别为

$$profit^1 = 900 \times 10^5 \frac{Q + p^{1*}}{2Q + p^{1*} + p^{2*}} - FC = 45625000 > 45000000$$
$$profit^2 = 700 \times 10^5 \frac{Q + p^{1*}}{2Q + p^{1*} + p^{2*}} - FC = 36375000 < 45000000$$

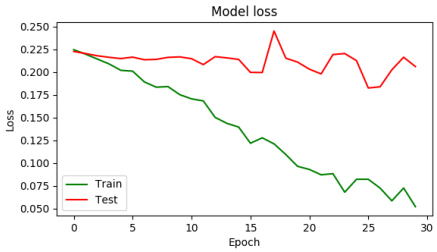
说明美的公司不会进入广告市场. 即此时真正的均衡广告出价应为 $(1990199.05, 0)$, 双方真正的利润分别为 $(82847094.99, -2847094.99)$, 虽然美的公司利润为负, 但是由于亏损额小于不变成本 $FC$, 因此美的公司会继续经营空调业务.

将上述过程循环执行, 最后能得出若美的公司不提高技术降低成本, 则由于连续的亏损, 其终究退出市场的结论.

# 六、 模型检验

第一问, 神经网络的 Accuracy 及 loss 如下图所示:


(d) Model Accuracy

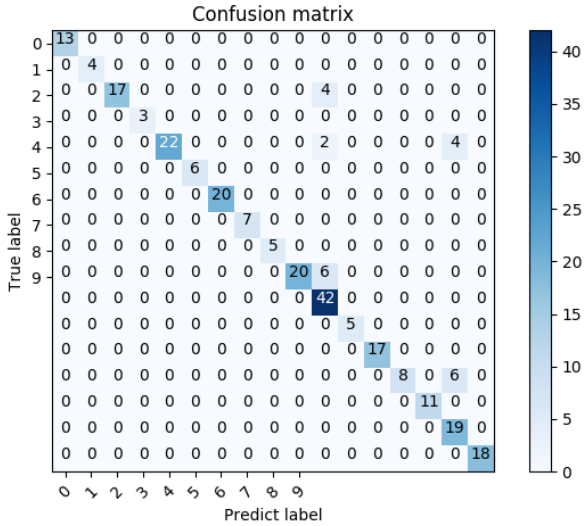



(e) Model Loss

这里我们选择的 loss 函数为 binary-crossentrophy 交叉熵损失函数, 即

$$loss = -\sum_{i=1}^{n} \hat{y}_i \log y_i + (1 - \hat{y}_i) \log(1 - \hat{y}_i)$$

混淆矩阵如下:

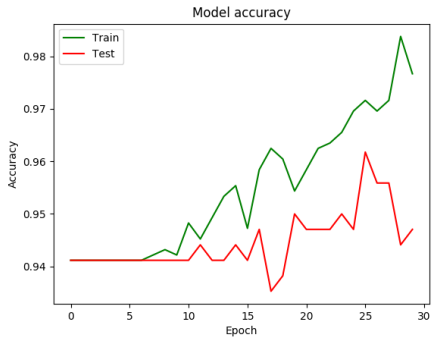



在评估 DeepFm 的模型性能中, 我们使用 Gini Normalization 作为性能指标. 基尼系数是一个分布不平衡程度的度量. 它被定义成大小在 0 到 1 之间的比值: 分子是均匀分布直线与洛伦兹曲线（下图中蓝色曲线）之间的面积, 分母是均匀分布直线下方的面积. 基尼系数越接近 1, 表示模型的预测能力越强.

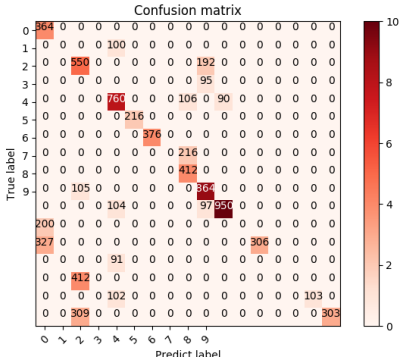
我们看出混淆矩阵中非 0 元素主要集中于主对角线, 这说明我们的模型命中率较好, 真实度较高.

第二问, 我们从附录 A 的检验报告可看出, 所有的系数均显著, 且经济意义正确, 时间序列平稳且系数高度显著.

第三问, 和第一问类似, 其 loss 和混淆矩阵如下图




(f) Model Accuracy




(g) Model Loss

可见其真实程度较好.

# 七、 模型评价

## 7.1 优点

- 神经网络模型准确率较高.
- relu 函数相比于 sigmoid 函数而言, 计算代价小, 速度快.
- 第二问第四问模型直观易懂, 经济学意义明显, 较好的反映了广告的供求关系.
- 第四问借助博弈论的理论, 将复杂的竞价交易过程简化成经济学模型.

## 7.2 缺点

- BP 神经网络训练能力同预测能力有时会存在矛盾, 也可能出现过拟合现象.
- 第二问采用多元线性回归, 检验其异方差性, 内生性等等会更完善.
- 某些线性回归的可决系数 $R^2$ 只有 50% 左右, 仍有提高的空间.
- 我们假设市场的需求不变, 其实可以尝试引入时间变量, 将需求设置为价格和时间的函数, 会更加的贴近实际.

# 参考文献


[1] 马洪宽. 博弈论. 同济大学出版社, 2015.

[2] 艾鹏强. 基于时序行为和标签关系的个性化新闻推荐系统研究. Master's thesis, 天津理工大学, 2016.

[3] 高鸿业 et al. 微观经济学. 北京: 中国人民大学出版社, 2007.

[4] 曼昆. 宏观经济学. 中国人民大学出版社, 2016.

[5] 李子奈, 潘文卿, et al. 计量经济学, volume 7. 高等教育出版社, 2000.

# 附录 A 第一问代码

```
import numpy as np
import tensorflow as tf

from time import time
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.metrics import roc_auc_score

class DeepFM(BaseEstimator, TransformerMixin):

def __init__(self, feature_size, field_size,
embedding_size=8, dropout_fm=[1.0, 1.0],
deep_layers=[32, 32], dropout_deep=[0.5, 0.5, 0.5],
deep_layer_activation=tf.nn.relu,
epoch=10, batch_size=256,
learning_rate=0.001, optimizer="adam",
batch_norm=0, batch_norm_decay=0.995,
verbose=False, random_seed=2016,
use_fm=True, use_deep=True,
loss_type="logloss", eval_metric=roc_auc_score,
l2_reg=0.0, greater_is_better=True):
assert (use_fm or use_deep)
assert loss_type in ["logloss", "mse"], \
"loss_type can be either 'logloss' for classification task or 'mse' for
    regression task"

self.feature_size = feature_size
self.field_size = field_size
self.embedding_size = embedding_size

self.dropout_fm = dropout_fm
self.deep_layers = deep_layers
self.dropout_dep = dropout_deep
self.deep_layers_activation = deep_layer_activation
self.use_fm = use_fm
self.use_deep = use_deep
```

```
self.l2_reg = l2_reg

self.epoch = epoch
self.batch_size = batch_size
self.learning_rate = learning_rate
self.optimizer_type = optimizer

self.batch_norm = batch_norm
self.batch_norm_decay = batch_norm_decay

self.verbose = verbose
self.random_seed = random_seed
self.loss_type = loss_type
self.eval_metric = eval_metric
self.greater_is_better = greater_is_better
self.train_result,self.valid_result = [],[]

self._init_graph()

def _init_graph(self):
self.graph = tf.Graph()
with self.graph.as_default():
tf.set_random_seed(self.random_seed)

self.feat_index = tf.placeholder(tf.int32,
shape=[None,None],
name='feat_index')
self.feat_value = tf.placeholder(tf.float32,
shape=[None,None],
name='feat_value')

self.label = tf.placeholder(tf.float32,shape=[None,1],name='label')
self.dropout_keep_fm =
    tf.placeholder(tf.float32,shape=[None],name='dropout_keep_fm')
self.dropout_keep_deep =
    tf.placeholder(tf.float32,shape=[None],name='dropout_deep_deep')
self.train_phase = tf.placeholder(tf.bool,name='train_phase')
```

```
self.weights = self._initialize_weights()

# model
self.embeddings =
    tf.nn.embedding_lookup(self.weights['feature_embeddings'],self.feat_index)
    # N * F * K
feat_value = tf.reshape(self.feat_value,shape=[-1,self.field_size,1])
self.embeddings = tf.multiply(self.embeddings,feat_value)


# first order term
self.y_first_order =
    tf.nn.embedding_lookup(self.weights['feature_bias'],self.feat_index)
self.y_first_order =
    tf.reduce_sum(tf.multiply(self.y_first_order,feat_value),2)
self.y_first_order = tf.nn.dropout(self.y_first_order,self.dropout_keep_fm[0])

# second order term
# sum-square-part
self.summed_features_emb = tf.reduce_sum(self.embeddings,1) # None * k
self.summed_features_emb_square = tf.square(self.summed_features_emb) # None *
    K

# squre-sum-part
self.squared_features_emb = tf.square(self.embeddings)
self.squared_sum_features_emb = tf.reduce_sum(self.squared_features_emb, 1) #
    None * K

#second order
self.y_second_order = 0.5 *
    tf.subtract(self.summed_features_emb_square,self.squared_sum_features_emb)
self.y_second_order =
    tf.nn.dropout(self.y_second_order,self.dropout_keep_fm[1])


# Deep component
```

```
self.y_deep = tf.reshape(self.embeddings,shape=[-1,self.field_size *
    self.embedding_size])
self.y_deep = tf.nn.dropout(self.y_deep,self.dropout_keep_deep[0])

for i in range(0,len(self.deep_layers)):
self.y_deep = tf.add(tf.matmul(self.y_deep,self.weights["layer_%d" %i]),
    self.weights["bias_%d"%i])
self.y_deep = self.deep_layers_activation(self.y_deep)
self.y_deep = tf.nn.dropout(self.y_deep,self.dropout_keep_deep[i+1])



#----DeepFM---------
if self.use_fm and self.use_deep:
concat_input = tf.concat([self.y_first_order, self.y_second_order,
    self.y_deep], axis=1)
elif self.use_fm:
concat_input = tf.concat([self.y_first_order, self.y_second_order], axis=1)
elif self.use_deep:
concat_input = self.y_deep

self.out =
    tf.add(tf.matmul(concat_input,self.weights['concat_projection']),self.weights['concat_bi

# loss
if self.loss_type == "logloss":
self.out = tf.nn.sigmoid(self.out)
self.loss = tf.losses.log_loss(self.label, self.out)
elif self.loss_type == "mse":
self.loss = tf.nn.l2_loss(tf.subtract(self.label, self.out))
# l2 regularization on weights
if self.l2_reg > 0:
self.loss += tf.contrib.layers.l2_regularizer(
self.l2_reg)(self.weights["concat_projection"])
if self.use_deep:
for i in range(len(self.deep_layers)):
self.loss += tf.contrib.layers.l2_regularizer(
self.l2_reg)(self.weights["layer_%d" % i])
```

```
if self.optimizer_type == "adam":
self.optimizer = tf.train.AdamOptimizer(learning_rate=self.learning_rate,
    beta1=0.9, beta2=0.999,
epsilon=1e-8).minimize(self.loss)
elif self.optimizer_type == "adagrad":
self.optimizer = tf.train.AdagradOptimizer(learning_rate=self.learning_rate,
initial_accumulator_value=1e-8).minimize(self.loss)
elif self.optimizer_type == "gd":
self.optimizer =
    tf.train.GradientDescentOptimizer(learning_rate=self.learning_rate).minimize(self.loss)
elif self.optimizer_type == "momentum":
self.optimizer = tf.train.MomentumOptimizer(learning_rate=self.learning_rate,
    momentum=0.95).minimize(
self.loss)


#init
self.saver = tf.train.Saver()
init = tf.global_variables_initializer()
self.sess = tf.Session()
self.sess.run(init)

# number of params
total_parameters = 0
for variable in self.weights.values():
shape = variable.get_shape()
variable_parameters = 1
for dim in shape:
variable_parameters *= dim.value
total_parameters += variable_parameters
if self.verbose > 0:
print("#params: %d" % total_parameters)
```

```
def _initialize_weights(self):
weights = dict()

#embeddings
weights['feature_embeddings'] = tf.Variable(
tf.random_normal([self.feature_size,self.embedding_size],0.0,0.01),
name='feature_embeddings')
weights['feature_bias'] =
    tf.Variable(tf.random_normal([self.feature_size,1],0.0,1.0),name='feature_bias')



#deep layers
num_layer = len(self.deep_layers)
input_size = self.field_size * self.embedding_size
glorot = np.sqrt(2.0/(input_size + self.deep_layers[0]))

weights['layer_0'] = tf.Variable(
np.random.normal(loc=0,scale=glorot,size=(input_size,self.deep_layers[0])),dtype=np.float32
)
weights['bias_0'] = tf.Variable(
np.random.normal(loc=0,scale=glorot,size=(1,self.deep_layers[0])),dtype=np.float32
)

for i in range(1,num_layer):
glorot = np.sqrt(2.0 / (self.deep_layers[i - 1] + self.deep_layers[i]))
weights["layer_%d" % i] = tf.Variable(
np.random.normal(loc=0, scale=glorot, size=(self.deep_layers[i - 1],
    self.deep_layers[i])),
dtype=np.float32) # layers[i-1] * layers[i]
weights["bias_%d" % i] = tf.Variable(
np.random.normal(loc=0, scale=glorot, size=(1, self.deep_layers[i])),
dtype=np.float32) # 1 * layer[i]

# final concat projection layer

if self.use_fm and self.use_deep:
input_size = self.field_size + self.embedding_size + self.deep_layers[-1]
```

```
elif self.use_fm:
input_size = self.field_size + self.embedding_size
elif self.use_deep:
input_size = self.deep_layers[-1]

glorot = np.sqrt(2.0/(input_size + 1))
weights['concat_projection'] =
    tf.Variable(np.random.normal(loc=0,scale=glorot,size=(input_size,1)),dtype=np.float32)
weights['concat_bias'] = tf.Variable(tf.constant(0.01),dtype=np.float32)

return weights

def get_batch(self,Xi,Xv,y,batch_size,index):
start = index * batch_size
end = (index + 1) * batch_size
end = end if end < len(y) else len(y)
return Xi[start:end],Xv[start:end],[[y_] for y_ in y[start:end]]

# shuffle three lists simutaneously
def shuffle_in_unison_scary(self, a, b, c):
rng_state = np.random.get_state()
np.random.shuffle(a)
np.random.set_state(rng_state)
np.random.shuffle(b)
np.random.set_state(rng_state)
np.random.shuffle(c)

def evaluate(self, Xi, Xv, y):
y_pred = self.predict(Xi, Xv)
return self.eval_metric(y, y_pred)

def predict(self, Xi, Xv):
# dummy y
dummy_y = [1] * len(Xi)
batch_index = 0
Xi_batch, Xv_batch, y_batch = self.get_batch(Xi, Xv, dummy_y, self.batch_size,
    batch_index)
```

```
y_pred = None
while len(Xi_batch) > 0:
num_batch = len(y_batch)
feed_dict = {self.feat_index: Xi_batch,
self.feat_value: Xv_batch,
self.label: y_batch,
self.dropout_keep_fm: [1.0] * len(self.dropout_fm),
self.dropout_keep_deep: [1.0] * len(self.dropout_dep),
self.train_phase: False}
batch_out = self.sess.run(self.out, feed_dict=feed_dict)

if batch_index == 0:
y_pred = np.reshape(batch_out, (num_batch,))
else:
y_pred = np.concatenate((y_pred, np.reshape(batch_out, (num_batch,))))

batch_index += 1
Xi_batch, Xv_batch, y_batch = self.get_batch(Xi, Xv, dummy_y, self.batch_size,
    batch_index)

return y_pred

def fit_on_batch(self,Xi,Xv,y):
feed_dict = {self.feat_index:Xi,
self.feat_value:Xv,
self.label:y,
self.dropout_keep_fm:self.dropout_fm,
self.dropout_keep_deep:self.dropout_dep,
self.train_phase:True}

loss,opt = self.sess.run([self.loss,self.optimizer],feed_dict=feed_dict)

return loss

def fit(self, Xi_train, Xv_train, y_train,
Xi_valid=None, Xv_valid=None, y_valid=None,
early_stopping=False, refit=False):
```

```
has_valid = Xv_valid is not None
for epoch in range(self.epoch):
t1 = time()
self.shuffle_in_unison_scary(Xi_train, Xv_train, y_train)
total_batch = int(len(y_train) / self.batch_size)
for i in range(total_batch):
Xi_batch, Xv_batch, y_batch = self.get_batch(Xi_train, Xv_train, y_train,
    self.batch_size, i)
self.fit_on_batch(Xi_batch, Xv_batch, y_batch)

# evaluate training and validation datasets
train_result = self.evaluate(Xi_train, Xv_train, y_train)
self.train_result.append(train_result)
if has_valid:
valid_result = self.evaluate(Xi_valid, Xv_valid, y_valid)
self.valid_result.append(valid_result)
if self.verbose > 0 and epoch % self.verbose == 0:
if has_valid:
print("[%d] train-result=%.4f, valid-result=%.4f [%.1f s]"
% (epoch + 1, train_result, valid_result, time() - t1))
else:
print("[%d] train-result=%.4f [%.1f s]"
% (epoch + 1, train_result, time() - t1))
if has_valid and early_stopping and
    self.training_termination(self.valid_result):
break

# fit a few more epoch on train+valid until result reaches the best_train_score
if has_valid and refit:
if self.greater_is_better:
best_valid_score = max(self.valid_result)
else:
best_valid_score = min(self.valid_result)
best_epoch = self.valid_result.index(best_valid_score)
best_train_score = self.train_result[best_epoch]
Xi_train = Xi_train + Xi_valid
Xv_train = Xv_train + Xv_valid
```

```
y_train = y_train + y_valid
for epoch in range(100):
self.shuffle_in_unison_scary(Xi_train, Xv_train, y_train)
total_batch = int(len(y_train) / self.batch_size)
for i in range(total_batch):
Xi_batch, Xv_batch, y_batch = self.get_batch(Xi_train, Xv_train, y_train,
self.batch_size, i)
self.fit_on_batch(Xi_batch, Xv_batch, y_batch)
# check
train_result = self.evaluate(Xi_train, Xv_train, y_train)
if abs(train_result - best_train_score) < 0.001 or \
(self.greater_is_better and train_result > best_train_score) or \
((not self.greater_is_better) and train_result < best_train_score):
break

def training_termination(self, valid_result):
if len(valid_result) > 5:
if self.greater_is_better:
if valid_result[-1] < valid_result[-2] and \
valid_result[-2] < valid_result[-3] and \
valid_result[-3] < valid_result[-4] and \
valid_result[-4] < valid_result[-5]:
return True
else:
if valid_result[-1] > valid_result[-2] and \
valid_result[-2] > valid_result[-3] and \
valid_result[-3] > valid_result[-4] and \
valid_result[-4] > valid_result[-5]:
return True
return False
```

# 附录 B 问题一完整结果

test loss: 0.1598327404413468

test accuracy: 0.9562594218131824

**Input** (’18:00-20:00’, ’35-44’, ’ 女’, ’ 本科及以上’, ’ 金融保险’, ’ 高’)

**Output** 衣着: 44.886% 金融/投资行业: 23.61% 化妆品/浴室用品: 11.862% 房地产: 8.7963% 个人用品: 2.2387% 家用电器: 2.1112%

**Input** (’0-18’, ’18:00-20:00’, ’ 低’, ’ 大专’, ’ 女’, ’ 教育’)

**Output** 娱乐及休闲: 92.5324% 交通: 3.7829% 家居用品: 1.3213% 药品: 0.9536% 家用电器: 0.5559% 个人用品: 0.2446%

**Input** (’18-24’, ’18:00-20:00’, ’ 低’, ’ 大专’, ’ 生活服务’, ’ 男’)

**Output** 交通: 99.6358% 娱乐及休闲: 0.146% 数码/电脑/办公用品: 0.0734% 房地产: 0.0548% 清洁用品: 0.0343% 酒类: 0.0177%

**Input** (’18-24’, ’20:00-22:00’, ’IT 电子通信’, ’ 中’, ’ 本科及以上’, ’ 男’)

**Output** 房地产: 62.9205% 数码/电脑/办公用品: 13.2506% 交通: 12.433% 清洁用品: 3.5259% 化妆品/浴室用品: 3.372% 酒类: 1.7673%

**Input** (’4:00-6:00’, ’65+’, ’ 低’, ’ 大专’, ’ 女’, ’ 无’)

**Output** 个人用品: 97.4525% 药品: 1.1369% 家居用品: 0.6533% 食品及饮料: 0.207% 化妆品/浴室用品: 0.1453% 衣着: 0.1231%

**Input** (’18-24’, ’22:00-0:00’, ’ 低’, ’ 大专’, ’ 女’, ’ 生活服务’)

**Output** 娱乐及休闲: 84.0095% 交通: 7.6668% 药品: 3.8912% 个人用品: 1.404% 家居用品: 1.3285% 清洁用品: 0.3355%

**Input** (’16:00-18:00’, ’18-24’, ’ 中’, ’ 本科及以上’, ’ 男’, ’ 金融保险’)

**Output** 酒类: 89.113% 食品及饮料: 3.0111% 交通: 2.8059% 化妆品/浴室用品: 1.2169% 房地产: 1.1838% 数码/电脑/办公用品: 0.7427%

**Input** (’10:00-12:00’, ’25-34’, ’ 中’, ’ 女’, ’ 文化体育娱乐’, ’ 本科及以上’)

**Output** 化妆品/浴室用品: 69.6046% 房地产: 9.5812% 金融/投资行业: 5.7738% 衣着: 5.4633% 清洁用品: 3.0178% 数码/电脑/办公用品: 1.6367%

**Input** (’10:00-12:00’, ’18-24’, ’IT 电子通信’, ’ 中’, ’ 女’, ’ 本科及以上’)

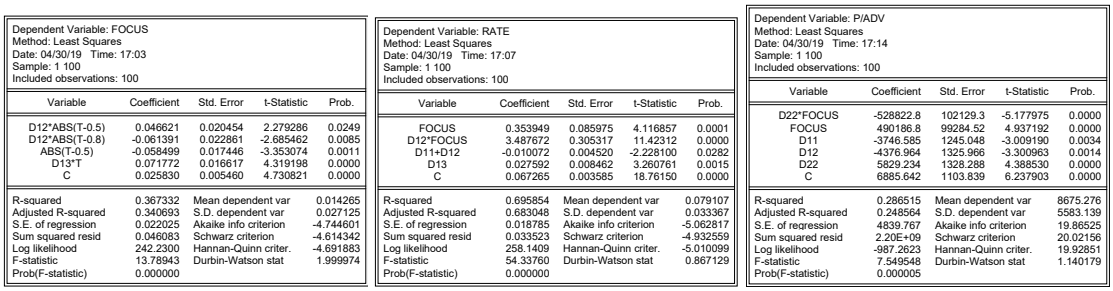
**Output** 清洁用品: 25.1317% 化妆品/浴室用品: 24.1783% 数码/电脑/办公用品: 23.2346% 房地产: 14.2259% 交通: 4.5675% 金融/投资行业: 2.8246%

**Input** (’10:00-12:00’, ’18-24’, ’ 低’, ’ 女’, ’ 本科及以上’, ’ 餐饮’)

**Output** 清洁用品: 50.2956% 化妆品/浴室用品: 28.4322% 个人用品: 4.6298% 数码/电脑/办公用品: 3.4407% 食品及饮料: 3.3067% 金融/投资行业: 2.8819%

# 附录 C 问题二实证分析 Eviews 多元线性回归报告

CCTV1 的三个回归方程的 Eviews 报告如下图所示:



Dependent Variable: FOCUS
Method: Least Squares
Date: 04/30/19 Time: 17:03
Sample: 1 100
Included observations: 100

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| D12*ABS(T-0.5) | 0.046621 | 0.020454 | 2.279286 | 0.0249 |
| D12*ABS(T-0.8) | -0.061391 | 0.022861 | -2.685462 | 0.0085 |
| ABS(T-0.5) | -0.058499 | 0.017446 | -3.353074 | 0.0011 |
| D13*T | 0.071772 | 0.016617 | 4.319198 | 0.0000 |
| C | 0.025830 | 0.005460 | 4.730821 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.367332 | Mean dependent var | 0.014265 |
| Adjusted R-squared | 0.340693 | S.D. dependent var | 0.027125 |
| S.E. of regression | 0.022025 | Akaike info criterion | -4.744601 |
| Sum squared resid | 0.046083 | Schwarz criterion | -4.614342 |
| Log likelihood | 242.2300 | Hannan-Quinn criter. | -4.691883 |
| F-statistic | 13.78943 | Durbin-Watson stat | 1.999974 |
| Prob(F-statistic) | 0.000000 | | |

Dependent Variable: RATE
Method: Least Squares
Date: 04/30/19 Time: 17:07
Sample: 1 100
Included observations: 100

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS | 0.353949 | 0.085975 | 4.116857 | 0.0001 |
| D12*FOCUS | 3.487672 | 0.305317 | 11.42312 | 0.0000 |
| D11+D12 | -0.010072 | 0.004520 | -2.228100 | 0.0282 |
| D13 | 0.027592 | 0.008462 | 3.260761 | 0.0015 |
| C | 0.067265 | 0.003585 | 18.76150 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.695854 | Mean dependent var | 0.079107 |
| Adjusted R-squared | 0.683048 | S.D. dependent var | 0.033367 |
| S.E. of regression | 0.018785 | Akaike info criterion | -5.062817 |
| Sum squared resid | 0.033523 | Schwarz criterion | -4.932559 |
| Log likelihood | 258.1409 | Hannan-Quinn criter. | -5.010099 |
| F-statistic | 54.33760 | Durbin-Watson stat | 0.867129 |
| Prob(F-statistic) | 0.000000 | | |

Dependent Variable: P/ADV
Method: Least Squares
Date: 04/30/19 Time: 17:14
Sample: 1 100
Included observations: 100

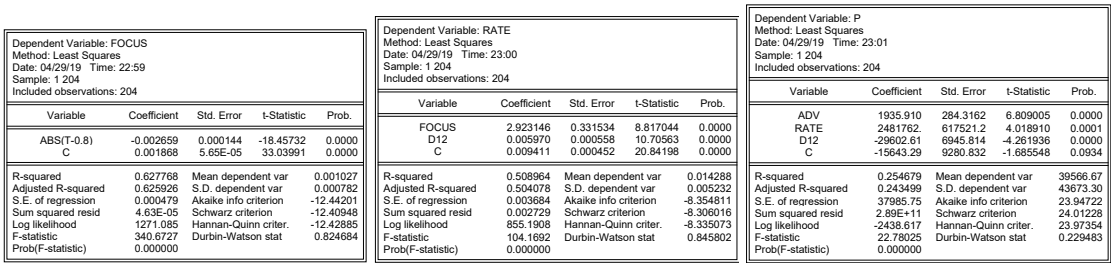
| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| D22*FOCUS | -528822.8 | 102129.3 | -5.177975 | 0.0000 |
| FOCUS | 490186.8 | 99284.52 | 4.937192 | 0.0000 |
| D11 | -3746.585 | 1245.048 | -3.009190 | 0.0034 |
| D12 | -4376.964 | 1325.966 | -3.300963 | 0.0014 |
| D22 | 5829.234 | 1328.288 | 4.388530 | 0.0000 |
| C | 6885.642 | 1103.839 | 6.237903 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.286515 | Mean dependent var | 8675.276 |
| Adjusted R-squared | 0.248564 | S.D. dependent var | 5583.139 |
| S.E. of regression | 4839.767 | Akaike info criterion | 19.86525 |
| Sum squared resid | 2.20E+09 | Schwarz criterion | 20.02156 |
| Log likelihood | -987.2623 | Hannan-Quinn criter. | 19.92851 |
| F-statistic | 7.549548 | Durbin-Watson stat | 1.140179 |
| Prob(F-statistic) | 0.000005 | | |

(h) $focus$ 函数　　(i) $rate$ 函数　　(j) $P/adv$ 函数

**图 4　CCTV1**

CCTV12 的三个回归方程的 Eviews 报告如下图所示:



Dependent Variable: FOCUS
Method: Least Squares
Date: 04/29/19 Time: 22:59
Sample: 1 204
Included observations: 204

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| ABS(T-0.8) | -0.002659 | 0.000144 | -18.45732 | 0.0000 |
| C | 0.001868 | 5.65E-05 | 33.03991 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.627768 | Mean dependent var | 0.001027 |
| Adjusted R-squared | 0.625926 | S.D. dependent var | 0.000782 |
| S.E. of regression | 0.000479 | Akaike info criterion | -12.44201 |
| Sum squared resid | 4.63E-05 | Schwarz criterion | -12.40948 |
| Log likelihood | 1271.085 | Hannan-Quinn criter. | -12.42885 |
| F-statistic | 340.6727 | Durbin-Watson stat | 0.824684 |
| Prob(F-statistic) | 0.000000 | | |

Dependent Variable: RATE
Method: Least Squares
Date: 04/29/19 Time: 23:00
Sample: 1 204
Included observations: 204

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS | 2.923146 | 0.331534 | 8.817044 | 0.0000 |
| D12 | 0.005970 | 0.000558 | 10.70563 | 0.0000 |
| C | 0.009411 | 0.000452 | 20.84198 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.508964 | Mean dependent var | 0.014288 |
| Adjusted R-squared | 0.504078 | S.D. dependent var | 0.005232 |
| S.E. of regression | 0.003684 | Akaike info criterion | -8.354811 |
| Sum squared resid | 0.002729 | Schwarz criterion | -8.306016 |
| Log likelihood | 855.1908 | Hannan-Quinn criter. | -8.335073 |
| F-statistic | 104.1692 | Durbin-Watson stat | 0.845802 |
| Prob(F-statistic) | 0.000000 | | |

Dependent Variable: P
Method: Least Squares
Date: 04/29/19 Time: 23:01
Sample: 1 204
Included observations: 204

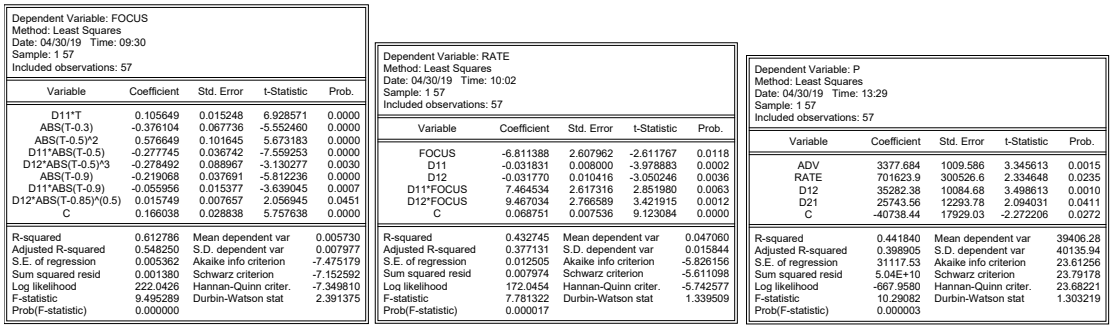
| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| ADV | 1935.910 | 284.3162 | 6.809005 | 0.0000 |
| RATE | 2481762. | 617521.2 | 4.018910 | 0.0001 |
| D12 | -29602.61 | 6945.814 | -4.261936 | 0.0000 |
| C | -15643.29 | 9280.832 | -1.685548 | 0.0934 |

| | | | |
|---|---|---|---|
| R-squared | 0.254679 | Mean dependent var | 39566.67 |
| Adjusted R-squared | 0.243499 | S.D. dependent var | 43673.30 |
| S.E. of regression | 37985.75 | Akaike info criterion | 23.94722 |
| Sum squared resid | 2.89E+11 | Schwarz criterion | 24.01228 |
| Log likelihood | -2438.617 | Hannan-Quinn criter. | 23.97354 |
| F-statistic | 22.78025 | Durbin-Watson stat | 0.229483 |
| Prob(F-statistic) | 0.000000 | | |

(a) $focus$ 函数　　(b) $rate$ 函数　　(c) $P$ 函数

**图 5　CCTV12**

湖南卫视的三个回归方程的 Eviews 报告如下图所示:



Dependent Variable: FOCUS
Method: Least Squares
Date: 04/30/19 Time: 09:30
Sample: 1 57
Included observations: 57

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| D11*T | 0.105649 | 0.015248 | 6.928571 | 0.0000 |
| ABS(T-0.3) | -0.376104 | 0.067736 | -5.552460 | 0.0000 |
| ABS(T-0.5)^2 | 0.576649 | 0.101645 | 5.673183 | 0.0000 |
| D11*ABS(T-0.5) | -0.277745 | 0.036742 | -7.559253 | 0.0000 |
| D12*ABS(T-0.5)^3 | -0.278492 | 0.088967 | -3.130277 | 0.0030 |
| ABS(T-0.9) | -0.219068 | 0.037691 | -5.812236 | 0.0000 |
| D11*ABS(T-0.9) | -0.055956 | 0.015377 | -3.639045 | 0.0007 |
| D12*ABS(T-0.85)^(0.5) | 0.015749 | 0.007657 | 2.056945 | 0.0451 |
| C | 0.166038 | 0.028838 | 5.757638 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.612786 | Mean dependent var | 0.005730 |
| Adjusted R-squared | 0.548250 | S.D. dependent var | 0.007977 |
| S.E. of regression | 0.005362 | Akaike info criterion | -7.475179 |
| Sum squared resid | 0.001380 | Schwarz criterion | -7.152592 |
| Log likelihood | 222.0426 | Hannan-Quinn criter. | -7.349810 |
| F-statistic | 9.495289 | Durbin-Watson stat | 2.391375 |
| Prob(F-statistic) | 0.000000 | | |

Dependent Variable: RATE
Method: Least Squares
Date: 04/30/19 Time: 10:02
Sample: 1 57
Included observations: 57

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| FOCUS | -6.811388 | 2.607962 | -2.611767 | 0.0118 |
| D11 | -0.031831 | 0.008000 | -3.978883 | 0.0002 |
| D12 | -0.031770 | 0.010416 | -3.050246 | 0.0036 |
| D11*FOCUS | 7.464534 | 2.617316 | 2.851980 | 0.0063 |
| D12*FOCUS | 9.467034 | 2.766589 | 3.421915 | 0.0012 |
| C | 0.068751 | 0.007536 | 9.123084 | 0.0000 |

| | | | |
|---|---|---|---|
| R-squared | 0.432745 | Mean dependent var | 0.047060 |
| Adjusted R-squared | 0.377131 | S.D. dependent var | 0.015844 |
| S.E. of regression | 0.012505 | Akaike info criterion | -5.826156 |
| Sum squared resid | 0.007974 | Schwarz criterion | -5.611098 |
| Log likelihood | 172.0454 | Hannan-Quinn criter. | -5.742577 |
| F-statistic | 7.781322 | Durbin-Watson stat | 1.339509 |
| Prob(F-statistic) | 0.000017 | | |

Dependent Variable: P
Method: Least Squares
Date: 04/30/19 Time: 13:29
Sample: 1 57
Included observations: 57

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
|---|---|---|---|---|
| ADV | 3377.684 | 1009.586 | 3.345613 | 0.0015 |
| RATE | 701623.9 | 300526.6 | 2.334648 | 0.0235 |
| D12 | 35282.38 | 10084.68 | 3.498613 | 0.0010 |
| D21 | 25743.56 | 12293.78 | 2.094031 | 0.0411 |
| C | -40738.44 | 17929.03 | -2.272206 | 0.0272 |

| | | | |
|---|---|---|---|
| R-squared | 0.441840 | Mean dependent var | 39406.28 |
| Adjusted R-squared | 0.398905 | S.D. dependent var | 40135.94 |
| S.E. of regression | 31117.53 | Akaike info criterion | 23.61256 |
| Sum squared resid | 5.04E+10 | Schwarz criterion | 23.79178 |
| Log likelihood | -667.9580 | Hannan-Quinn criter. | 23.68221 |
| F-statistic | 10.29082 | Durbin-Watson stat | 1.303219 |
| Prob(F-statistic) | 0.000003 | | |

(a) $focus$ 函数　　(b) $rate$ 函数　　(c) $P$ 函数

**图 6　湖南卫视**

# 附录 D 第二问代码

```
a=[...];% 这里是个收视率数据448, 比较长
s=8;%周期大小
n=24;%预测数量
m1=length(a);
for i=s+1:m1
   y(i-s)=a(i)-a(i-s);
end
w=diff(y);
%w=y;
m2=length(w);
k=0;
for i =0:3
   for j =0:3
      if i==0 &&j==0
         continue
      elseif i==0
         model=arima('MALags',1:j,'Constant',0);
      elseif j==0
         model=arima('ARLags',1:i,'Constant',0);
      else
         model=arima('ARLags',1:i,'MALags',1:j,'Constant',0);
      end
      k=k+1;R(k)=i;M(k)=j;
      [EstMd,EstParamConv,logL,info]=estimate(model,w');
      numParams=sum(any(EstParamConv));
      [aic(k),bic(k)]=aicbic(logL,numParams,m2);
   end
end
fprintf('R,M,AIC,的对应值如下BIC');
check=[R',M',aic',bic']
r=input('请输入阶数R=');m=input('请输入阶数M=');
model=arima('ARLags',1:r,'MALags',1:m,'Constant',0);
[EstMd,EstParamConv,logL,info]=estimate(model,w');
w_forecast=forecast(EstMd,n,'Y0',w');
yhat=y(end)+cumsum(w_forecast);
```

```
for j=1:n
   a(m1+j)=yhat(j)+a(m1+j-s);
   %a(m1+j)=w_forecast(j)+a(m1+j-s);
end
ahat=a(m1+1:end)
```

# 附录 E 第三问代码

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <map>
#include <cstdlib>
#include <set>
#include <ctime>

using namespace std;

map<int, int>ma;
queue<int>w, ww;
set<int>st;
int ml;
FILE *user, *ad, *ts;
double sigmaAll, sigmaOne[50];

const int nad = 100000;
const int firstsame = 50;
const int firstad = 15;
const int nuser = 5000;
const int del = 10;

struct u
{
   int id;
   int a[nad + 10];
   u *next;
   u()
   {
      id = 0;
      next = NULL;
      memset(a, 0, sizeof(a));
   }
}*rt, *tl;
```

```
struct uu
{
   int iduser;
   long long z;
   friend bool operator < (const uu &aa, const uu &bb)
   {
      return aa.z > bb.z;
   }
};

int type[nad+10];
map<int, u*>mmp;
priority_queue<uu>q;

struct u3
{
   int idad;
   double z;
   friend bool operator < (const u3 &aa, const u3 &bb)
   {
      if(aa.z==bb.z)
         return sigmaOne[type[aa.idad]]>sigmaOne[type[bb.idad]];
      return aa.z > bb.z;
   }
}c[nad + 10];

struct u4
{
   int id;
   int rk;
   friend bool operator < (const u4 &aa, const u4 &bb)
   {
      return aa.rk > bb.rk;
   }
}c2[500000];
```

```
priority_queue<u3>qq;

void append()
{
   u *nd = new u();
   int aa, bb, cc, dd, ee;
   fscanf(user, "%d%d%d", &aa, &bb, &cc);
   nd->id = bb;
   mmp[bb] = nd;
   fscanf(user, "%d", &cc);
   while (cc != -1)
   {
      nd->a[ma[cc]]++;
      sigmaOne[ type[ma[cc]]]+=1.0;
      sigmaAll += 1.0;
      fscanf(user, "%d", &cc);
   }
   if (rt == NULL)
   {
      rt = nd;
   }
   else
      tl->next = nd;
   tl = nd;
}

long long getSameValue(u *aa, u* bb)
{
   long long pr = 0;
   for (int i = 0;i < nad;i++)
      pr += (aa->a[i] - bb->a[i])*(aa->a[i] - bb->a[i]);
   return -pr;
}

void compare(u *aa)
{
   u *now = rt;
```

```
   while (!q.empty())
      q.pop();
   while (now != NULL)
   {
      uu newuu;
      newuu.iduser = now->id;
      newuu.z = getSameValue(now, aa);
      now = now->next;
      q.push(newuu);
      while (q.size() > firstsame)
         q.pop();
   }
}

bool g()
{
   st.clear();
   while (!qq.empty())
   {
      st.insert(qq.top().idad);
      qq.pop();
   }
   while (!ww.empty())
   {
      int p = ww.front();
      if (st.count(ww.front()) > 0)
         return true;
      ww.pop();
   }
   return false;
}



int main()
{
   long count = 0;
```

```
user = fopen("userfinal.txt", "r");
ad = fopen("ad5.txt", "r");
ts = fopen("usertestfinal.txt", "r");
for (int i = 1;i <= nad;i++)
{
   int aa, bb, cc;
   fscanf(ad, "%d%d%d", &aa, &bb, &cc);
   ma[bb] = ++ml;
   type[ml] = cc;
}
for (int i = 1;i <= nuser;i++)
{
   append();
}
for (int i = 1;i <= ml;i++)
   c[i].idad = i;
double cg = 0, zs = 0;
while (1)
{
   ++count;
   u *nd = new u();
   int aa, bb, cc, dd, ee;
   fscanf(ts, "%d%d%d", &aa, &bb, &cc);
   nd->id = bb;
   while (!w.empty()) w.pop();
   while (!ww.empty()) ww.pop();
   int bh = 0;
   fscanf(ts, "%d", &cc);
   while (cc != -1)
   {
      bh++;
      c2[bh].id = cc;
      c2[bh].rk = rand();
      fscanf(ts, "%d", &cc);
   }
   for (int i = 1;i <= bh;i++)
   if (i <= del)
```

```
      ww.push(ma[c2[i].id]);
   else
      (nd->a[ma[c2[i].id]])++;
   compare(nd);
   for (int i = 1;i <= ml;i++)
      c[i].z = 0;
   while (!q.empty())
   {
      int id = q.top().iduser;
      int pr = 0;
      for (int j = 1;j <= ml;j++)
         pr += (mmp[id]->a[j] > 0);
      u* uid = mmp[id];
      q.pop();
      for (int i = 1;i <= ml;i++)
         if (uid->a[i] > 0)
         {
            c[i].z+=1;
         }
   }
   for (int i = 1;i <= ml;i++)
   {
      qq.push(c[i]);
      while (qq.size() > firstad)
         qq.pop();
   }
   if (g())
   cg += 1.0;
   zs += 1.0;
   for (int i = 1;i <= del;i++)
      (nd->a[ma[c2[i].id]])++;
   u *p = mmp[nd->id];
   if (p == NULL)
   {
      tl->next = nd;
      tl = nd;
      mmp[nd->id] = nd;
```

```
        for (int i = 1;i <= ml;i++)
        {
            sigmaAll += nd->a[i];
            sigmaOne[type[i]] += nd->a[i];
        }
    }
    else
    {
        for (int i = 1;i <= ml;i++)
        {
            sigmaAll -= p->a[i];
            sigmaOne[type[i]] -= p->a[i];
            sigmaAll += nd->a[i];
            sigmaOne[type[i]] += nd->a[i];
            p->a[i] = nd->a[i];
        }
    }
    printf("%d: %lf\n",count , cg / zs);
  }
}
```

# 附录 F 第三问完整测试数据

| period | 个人用品 | 互联网 | 交通 | 农业 | 化妆品/浴室用品 | 商业/服务性行业 | 娱乐及休闲 | 家居用品 | 家用电器 | 房地产 | 数码/电脑/办公 | 清洁用品 | 药品 | 衣着 | 酒类 | 金融/投资行业 | 食品及饮料 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -0.0370 | -0.0007 | -0.0609 | 0.0159 | -0.0396 | 0.0286 | -0.0134 | -0.0497 | 0.0096 | -0.0239 | 0.0092 | -0.0050 | 0.0523 | 0.0053 | 0.0233 | -0.0385 | -0.0354 |
| 2 | -0.0101 | -0.0165 | 0.0504 | 0.0418 | 0.0046 | 0.0375 | -0.0337 | -0.0499 | -0.0473 | 0.0318 | 0.0369 | 0.0057 | -0.0303 | 0.0347 | 0.0260 | 0.0141 | 0.0022 |
| 3 | 0.0066 | -0.0851 | -0.0619 | 0.0466 | -0.0023 | 0.0172 | -0.0163 | 3.9999 | 0.0096 | 0.0007 | -0.0444 | -0.0785 | 0.0038 | -0.0023 | -0.0023 | 0.0077 | -0.0480 |
| 4 | -0.0126 | -0.0343 | 0.0129 | 0.0145 | 0.0391 | 0.0084 | 0.0262 | 3.0807 | -0.0282 | 0.0113 | -0.0128 | -0.0082 | 0.0037 | 0.0123 | -0.0182 | -0.0180 | -0.0775 |
| 5 | 0.0673 | -0.0373 | 0.0194 | -0.0088 | -0.0065 | 0.0012 | 0.0118 | 2.0733 | 0.0194 | -0.0762 | 0.0236 | -0.0032 | -0.0131 | 0.0368 | 0.0345 | -0.0146 | 0.0276 |
| 6 | -0.0297 | 0.0310 | 0.0754 | 0.0009 | -0.0151 | -0.0537 | -0.0147 | 0.9960 | 0.0010 | -0.0163 | -0.0118 | -0.0199 | 0.0209 | -0.0635 | 0.0044 | 0.0211 | 0.0382 |
| 7 | -0.0045 | -0.0248 | -0.0305 | -0.0496 | 0.0329 | -0.1007 | -0.0586 | 3.0215 | 0.0406 | -0.0019 | 0.0293 | 0.0349 | 0.0094 | 0.0325 | 0.0183 | 0.0434 | -0.0545 |
| 8 | -0.0413 | -0.0552 | -0.0254 | 0.0231 | 0.0133 | 0.0489 | -0.0516 | -0.0343 | 0.0538 | 0.0191 | -0.0987 | 0.0011 | 0.0422 | -0.0045 | -0.0409 | -0.0120 | 0.0194 |
| 9 | -0.0497 | 0.0590 | -0.0195 | -0.0427 | -0.0129 | -0.0112 | -0.0111 | 1.0599 | 0.0180 | 0.0607 | 0.0090 | -0.0015 | 0.0012 | -0.0443 | 0.0439 | 0.0066 | 0.0314 |
| 10 | 0.0551 | 0.0397 | -0.0438 | 0.0002 | -0.0271 | -0.0129 | 0.0210 | 1.0280 | 0.0019 | 0.0867 | 0.0206 | 0.0111 | -0.0214 | 0.0004 | 0.0159 | 0.0131 | 0.0005 |
| 11 | -0.0256 | -0.0347 | 0.0031 | 0.0198 | -0.0424 | -0.0167 | -0.0355 | 3.0173 | -0.0198 | 0.0156 | -0.0016 | 0.0736 | 0.0019 | 0.0247 | -0.0193 | 0.0119 | -0.0146 |
| 12 | 0.0334 | 0.0127 | -0.0074 | -0.0106 | 0.0273 | -0.0403 | 0.0380 | 1.9817 | 0.0523 | 0.0116 | 0.0178 | -0.0100 | -0.0209 | 0.0145 | 0.0699 | 0.0436 | 0.0431 |
| **sum** | **-0.0481** | **-0.1463** | **-0.0881** | **0.0511** | **-0.0286** | **-0.0937** | **-0.1380** | **20.1244** | **0.1110** | **0.1191** | **-0.0228** | **0.0001** | **0.0496** | **0.0466** | **0.1554** | **0.0783** | **-0.0675** |

# 附录 G 第四问 Matlab 代码

```
clear
clc
p1=0;
p2=0;
for i=1:10
p1=sqrt(900*10^5*(50000+p2)/1.03)-p2-100000;
p2=sqrt(700*10^5*(50000+p1)/1.03)-p1-100000;
fprintf('%d & %f & %f\\\\ \n',i,p1,p2)
end
p1=1990199.042875;
p2=0;
vpa(900*10^5*(p1+50000)/(p1+p2+100000)-5000000)
vpa(900*10^5*(p2+50000)/(p1+p2+100000)-5000000)
```