本科生课程论文

论文题目：“互联网+”时代的出租车资源配置

院　　系：计算机科学与技术学院

课　　程：数学建模

成　　员：耿志超 16300720039

周启辉 16300720005

杨希希 16307130365

祁佳薇 16307130293

日　　期：2019 年 6 月 10 日

“互联网+”时代的出租车资源配置

摘 要

“打车难”一直是近年来备受社会关注的问题，随着“互联网+”时代的到来，建立在移动互联网基础上的各类打车软件实现了乘客与司机之间消息的互通，已经成为大家生活中必备的工具。然而以打车软件代表的出租车市场中仍存在资源配置问题，如高峰时段乘客打车后出租车接单时间长甚至打不到车，而空闲时段出租车空驶率高；偏远位置的乘客很难通过叫车软件叫到车，并有可能遭到拒载。本文通过量化不同时空出租车资源的供求匹配程度，并根据此匹配程度制定补贴方案来改善出租车资源的供求关系，令不同时间和地点中的供求都达到平衡，从而缓解乘客打车难、减少司机空驶率、甚至增加本打车软件在市场中的竞争力。

针对问题一，根据从滴滴快的智能出行平台搜集到的 2016 年 8 月 6 日北京市 0 点至 23 点出租车分布和乘客需求数据，绘制图像，观察得出需求高峰期发生在早上 8 点和晚上 19 点左右，相应地供给也随之变化。通过创新性地选取空驶率、人均响应车数和平均等待时间三个指标，我们建立了供求匹配模型。从供给和需求的角度出发，求得指标理想值；然后根据实际数据画出时空分布图，得出北京市总体出租车资源供求匹配较为平衡，其中高峰时段优于常规时段，郊区优于市区。

针对问题二，我们分析了各公司的出租车补贴方案是否对“缓解打车难”有帮助。在绘制了滴滴和快的两个公司在不同时间补贴方案的图之后，我们总结两公司的补贴策略。提出了一个可以计算补贴与打车难环节程度之间关系的模型。通过公司对乘客的补贴金额和对司机的补贴金额两个指标，我们可以计算出接客半径，居民出行工具选择中采用出租车的比例和出租车的出车率这三个指标。最后通过期望等待时间和人均响应车数得到打车难环节程度。我们发现滴滴和快的都可以在一定程度上缓解打车难问题，但是缓解程度并不理想。

针对问题三，一个合理的补贴政策首先应该一定程度上缓解打车难的问题，所以沿用问题二中提出的缓解程度衡量指标 s，将 s>0 作为约束限制条件；在此基础上我们综合考虑了净收入、成交量、顾客满意度、补贴政策等因素，得出打车软件公司利润函数，其中顾客满意度是关于等待时间的函数，由已有数据拟合一个浅层神经网络来表示，问题转换为在约束下最大化利润函数，可求得合理补贴政策。

综上所述，本文通过建立供求匹配模型、缓解程度判断模型和利润最大化模型，对北京市出租车资源的供求匹配程度和现有补贴方案进行了分析评价，并设计了合理的补贴方案。所建模型具有较好的实用性，对于今后实际出租车资源配置问题解决具有一定参考价值。此外，本课题制定的评估规则不只局限于打车，也可推广至其它供求关系的衡量中，例如“外卖接单”时空分布的供求问题中，从而在创造利润的同时最大化供求双方的满意程度。

关键词：出租车资源配置 供求匹配模型 时空分布 补贴政策

一、问题提出

1.1 背景

出租车是城市居民出行的重要交通工具之一，但随着城市人口密度和私家车密度的增加，“打车难”已成为社会热点问题，包括出租车绝对数量供给不足、信息不对称导致供需时空分布不匹配等。随着“互联网+”时代的到来，多家公司例如滴滴快的等依托移动互联网建立了打车软件服务平台，实现了乘客与出租车司机之间的信息互通，同时推出了多种出租车的补贴方案，刺激了消费群体和出租车司机的积极性。政府和打车软件公司在决策过程中都会遇到如下一系列问题，这些问题就是本文探讨的重点。

1.2 问题提出

(1) 根据搜集到的相关数据，建立合理的指标，分析不同时空出租车资源的“供求匹配”程度。
(2) 分析各公司的出租车补贴方案是否对“缓解打车难”有帮助。
(3) 尝试设计新的补贴方案，满足在缓解打车难问题的同时使打车软件平台利润最大化。

二、问题分析

2.1 问题一的分析

该问题要求建立合理的指标，并分析不同时空出租车资源的“供求匹配”程度。首先根据滴滴快的智能出行平台搜集了 2016 年 8 月 6 日北京市 0 点至 23 点出租车的分布情况和乘客的需求情况。对获取的数据进行可视化处理，做出反映出租车位置和乘客位置的分布图和各时点均值变化图，总体上感受了需求高峰低谷和北京市打车难现状。在此基础上，建立匹配模型。考虑到万人拥有量是一个静态指标，与我们研究的供求关系随时空变化而动态过程不符；无法获取用于计算里程利用率的数据；所以最终我们选取了三个指标——空驶率、人均响应车数和平均等待时间来定量衡量打车难的程度。分别计算三个指标的理想值，再画出空间上、时间上指标的实际值，通过比较得出北京市打车的难易程度。

2.2 问题二的分析

该问题要求分析各公司的出租车补贴方案是否对“缓解打车难”有帮助，我们首先描绘出滴滴和快的两个公司在不同时间补贴方案的图，以快的打车为例，计算出公司对乘客的补贴金额λ_1和对司机的补贴金额λ_2，通过意愿半径 R，居民出行工具选择中采用出租车的比例 p 和出租车的出车率α这三个指标，分别对未使用补贴方案及使用补贴方案两种情况进行分析对比。为了分析出租车的补贴政策是否缓解了打车难的问题，我们可以利用问题一中提出的两个指标：期望等待时间 t 和人均响应车数η来衡量。同时，我们提出使用公式：$s = \lg\frac{t^*}{t} + lg\frac{\eta}{\eta^*}$，这个公式综合考虑了期望等待时间 t 和人均响应车数η的变化，来判断补贴方案对于打车难的缓解程度。

2.3 问题三的分析

该问题要求设计新的补贴方案，我们从缓解打车难问题和利润最大化两个方面考虑。针对打车难问题，问题二中提出了基于期望等待时间 t 和人均响应车数η的缓解程度衡量指标$s = \lg\frac{t^*}{t} + lg\frac{\eta}{\eta^*}$，在问题三中作为约束限制条件，需保证 s>0；针对利润问题，我们提出了综合考虑接单数量、每单利润和对司机、乘客补贴的利润函数 $\gamma = n\sigma p\delta * (\beta - \lambda_1 - \lambda_2)$，运用网格法可求解利润最大值，和相应的最优补贴政策。

三、模型假设

1、出租车司机收入按正常打表计算，不考虑消费者额外给的小费。
2、司机认为利益受损失不会接单，即不会前往顾客所在地。
3、司机一旦到达乘客所在处就表示一定接单。
4、考虑现实生活中的拒单、拼车等实际情况。
5、顾客按单计算，即两人一起拼车记为一单。
6、假设司机和等车乘客按二维正态分布存在于在一个城市中。
7、假设使用打车软件打车的情况可以估计所有的打车情况。
8、假设乘客和出租车司机会因补贴政策的驱使而倾向于使用打车软件。
9、不考虑突发情况，极端自然状况导致的绕行和停车。

四、符号说明

符号	说明
N1	出租车供给量
N2	居民需要的出租车次数
α	出租车的出车率
K	空驶率
σ	居民平均每天出行次数
λ_1	乘客补贴
λ_2	司机补贴
η	人均可应答车辆数
R	接客半径
t	等待时间
p	居民采用出租车出行比例
s	打车难缓解程度
n	居民人数
m	接单数量
δ	选择出租车时使用我们软件的比例
f	乘客满意度
c	接客人数最大值
N	北京市出租车总数
L	北京市道路总长度
T	出租车平均运营时间
v	出租车平均车速度
Si	第 i 个行人的点
Di	第 i 个出租车的点
X	偏离程度

五、模型的建立与求解

5.1 问题一的建模与求解

根据题目要求，首先从数据平台获取原始数据，预处理后通过绘图大体感知供需匹配现状，然后确立定量衡量指标，通过实际值与理想值的比较得出结论。

5.1.1 数据获取与预处理

基于“苍穹——滴滴快的智能出行平台”编写爬虫程序，获得了 2016 年 8 月 6 日至 12 日 7 天北京市全市出租车数量分布、乘客打车需求、等待时间、乘客满意度、交易额等 5 项数据作为样本。每项数据均为每小时采样一次，每日 0 点至 23 点 24 次采样，每次采样选取全市约 500 个采样点。

因为平台数据仅公开至 2016 年，近两年随着网约车进一步普及，出租车数量、居民出行方式等都发生了改变，所以与 2019 年现状不完全符合。此外还应注意到我们的数据来源仅为占市场主要份额的滴滴快的公司，不能代替全市出租车的运营情况。

对原始数据的处理包括首先去除坏值（经纬度偏差较大的值和信息异常的值），结合 echarts 调用百度地图 api，利用处理后的数据绘制某一小时出租车分布、用户需求的二维散点图，散点由经纬度定位，点阵的大小反映出租车数量或乘客需求量的多少。

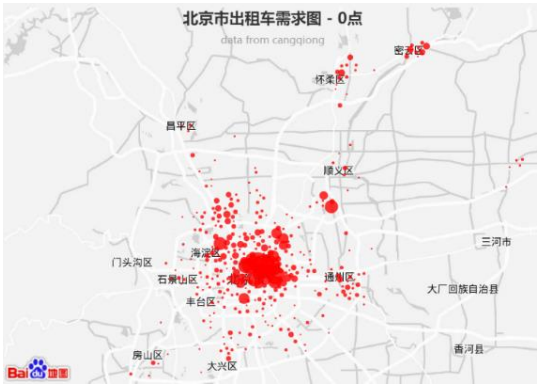


图 1-1 北京市 0 点出租车需求分布图

图 1-2 北京市 0 点出租车供应分布图

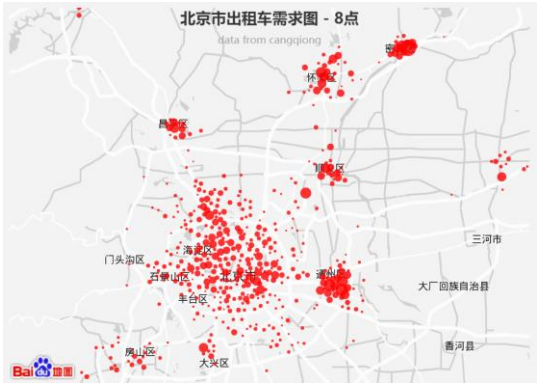


图 1-3 北京市 8 点出租车需求分布图

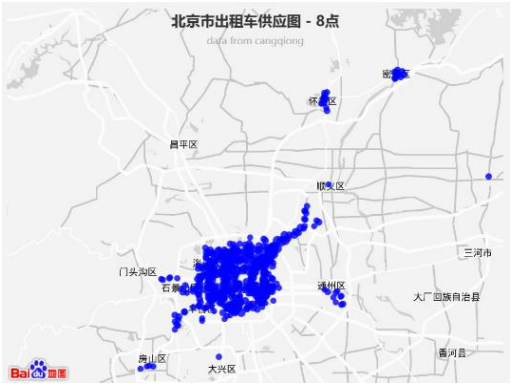


图 1-4 北京市 8 点出租车供应分布图

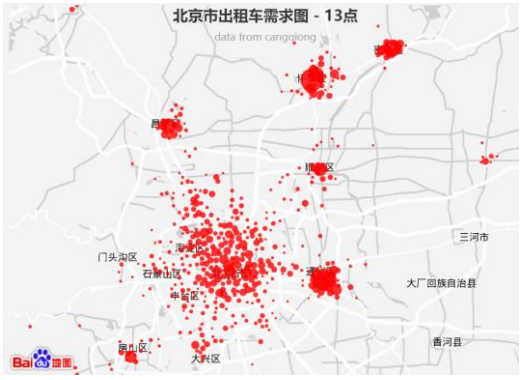


图 1-5 北京市 13 点出租车需求分布图

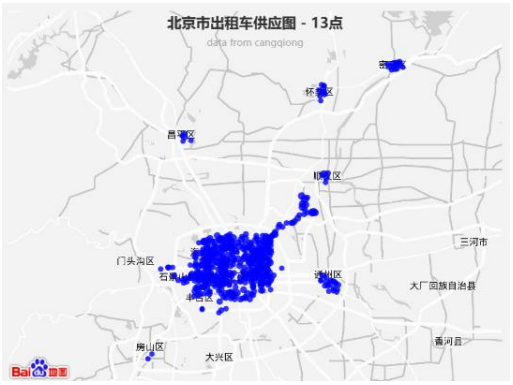


图 1-6 北京市 13 点出租车供应分布图

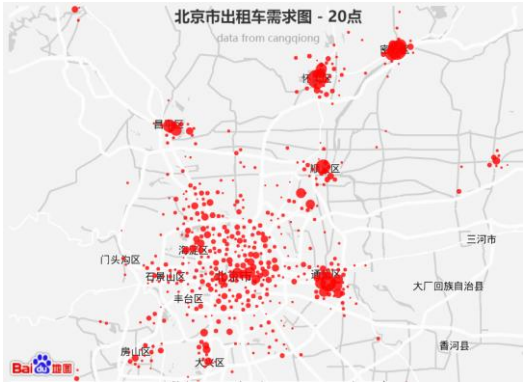


图 1-7 北京市 20 点出租车需求分布图

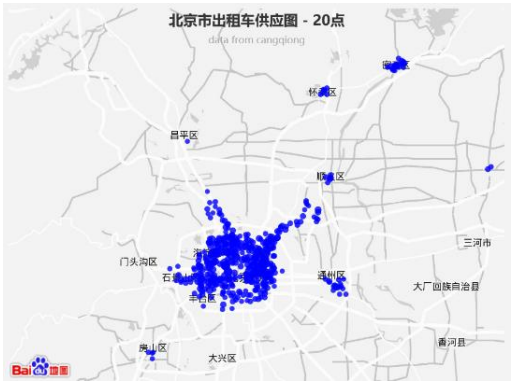


图 1-8 北京市 20 点出租车供应分布图

图 1-1-图 1-8 给出了 0 点，8 点，13 点和 20 点的北京市乘客需求和出租车供应空间分布图。首先可以看到各个时间都有需求，也都有出租车分布。其次从空间分布上看，市中心与郊区相比，市中心总体需求量和供应量较大，虽然北京市有超过一半的人口居住在五环以外，这部分人也有到市中心上班或者其它日常出行需求，但是出租车的分布相对需求较少，较多地往返后可能仍停留在中心。另外市中心的分布在五环内较为平均，呈弥漫性分布；而郊区则较为集中分布在昌平区、通州区等区内中心。从时间分布上看，0 点代表的低峰时段需求和分布都较少，8 点、13 点、20 点代表的高峰时段明显较多。所以由上面几张图我们基本可以看出出租车在时空分布上存在不均匀的情况，有些供应没有随着需求变化，可能由于出租车司机有拒载的情况，使得司机根据距离等挑选客人。

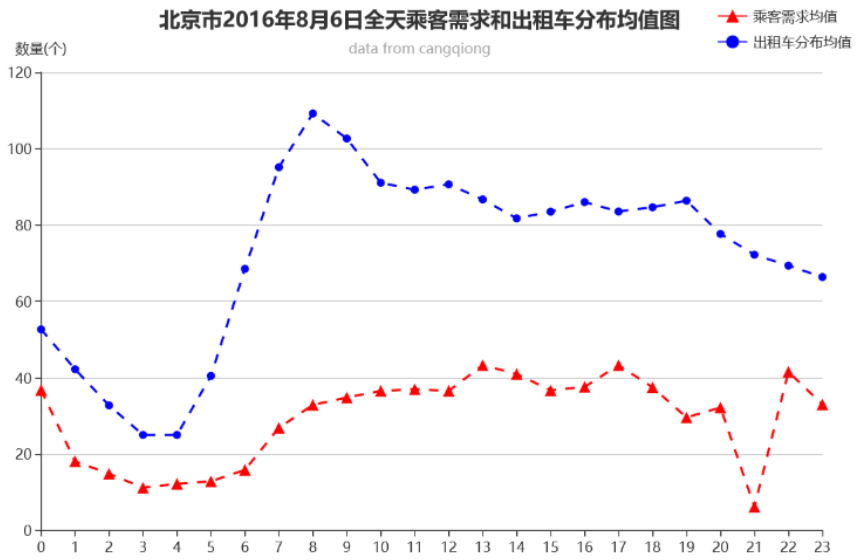


图 2 北京市 2016 年 8 月 6 日全天乘客需求和出租车分布均值图

为了更好地观察各个时段出租车和乘客的匹配情况，我们按每个时点计算了出租车分布和乘客需求的均值，从图中可以明显看出，各时间段北京市平均出租车运行数量大于乘客平均需求量，尤其是高峰时刻，大约相差 2 倍左右。因此总体而言，不会出现“打车难”的问题。

单看需求曲线，凌晨 1-6 点是乘客需求量的低谷，平均稳定在 15 个左右，此时出行需求较少；从早上 7 点开始需求逐渐上升，10 点至下午 17 点维持较高水平，在 40 个左右，13 点和 17 点出现两个小高峰，这与我们的预期结果有一定差异，猜测原因是：①除了通勤需求，其它类型需求也占很大的比例，根据《2017 年北京交通发展年报》，中心城工作日日均出行总量为 2666 万人次，而通勤出行量为 1779 万人次，还有一半其它类型的需求；②通勤选择出租车比例并不高，根据年报数据，该比例只占到 2.9%，其它轨道、公交、小汽车、班车、自行车等占了绝大部分，因此通勤需求有影响、但是没有决定曲线趋势变化。18-21 点出现一个需求地小低谷，分析原因可能是第一批下班人群已经到家，大多数人正处于晚饭时间；而 22 点出现了回升，至次日凌晨 0 点需求水平一直都较高，考虑到当下社会热议的加班现象，可能是第二批下班人群，以及夜生活丰富的人。

单看供应曲线，即出租车数量变化，可以看出与需求有相似之处。在凌晨 3-4 点的时候分布最少，约为 25 辆左右，最高点出现在 8 点，平均有 110 辆出租车在运行；8-10 点出现回落，整个白天的需求维持在 85 上下；19 点以后开始持续下降，一直到次日凌晨，没有看到针对 22 点需求的小高峰。供应的变化可能是为了响应需求，例如早上的通勤需求和白天的日常需求；也可能本来大多数出租车司机工作的时间是在白天、开始是在早上 7、8 点，熬夜干活挣钱的现象较少。

5.1.2 衡量指标确立

虽然平均而言不会出现“打车难”的问题，但是结合平时生活经验以及上图，可以看出出租车和乘客需求的分布存在不均衡的现象。有的区域非常集中，而个别乘客点十分孤立；低峰时刻需求较容易满足，而高峰时刻可能出现“一车难求”的现象。为此我们建立模型来定量衡量“打车难”的程度。

“供应匹配”分为三种情况：供求平衡、供大于求、供小于求。为了衡量不同时空出租车资源的“供求匹配”程度，我们确立了空驶率、人均响应车数和平均等待时间三个指标。

1、空驶率 K

出租车空驶率分为时间上和空间上的空驶率，时间上的空驶率是指一定时间内出租车空驶时间与总行驶时间的比值，空间上的空驶率是指一定时间内出租车空驶里程与总行驶里程的比值。结合我们获取的数据进行适当的定义改写，公式表示如下：

$$空驶率\ K = \frac{总车辆-需求量}{总车辆} \times 100\%$$

这一指标反映了出租车的载客效率，空驶率越低，说明车辆行驶中载客比例越高，对于打车的乘客来说可选的车辆越少，即供求关系越紧张。

2、人均响应车数η

人均响应车数是指一定范围内总响应数量与总需求数量的比值，公式表示如下：

$$人均响应车数\ \eta = \frac{一定范围内响应量总和}{需求量总和}$$

这一指标用于衡量出租车与用户在数量上的供求关系，当人均响应车数等于 1 时，刚好每位乘客都能打到车，供求平衡；大于 1 时，供大于求；小于 1 时，供不应求。

3、平均等待时间 t

平均等待时间是指从乘客下单，经过出租车司机接单和接单后前往接客，到乘客上车所经过的时长。经过分析，该指标和空驶率之间应该有联系。空驶率越高，平均等待时间越短，因为可接单的人数多；空驶率越低，平均等待时间越长。

5.1.3 模型建立与求解

1、空驶率

首先确定空驶率的理想值：

1）从供给角度

假设以下参数：

N1 —— 出租车供给量

K —— 空驶率

N —— 某地区出租车的数

α —— 出车率

c —— 日均接客人数

则出租车供应量为：

$$N1 = (1 - K)cN\alpha$$

2）从需求角度

假设以下参数：

N2 —— 居民需要的出租车次数

n —— 居民人数

σ —— 居民平均每天出行次数

p —— 居民出行工具选择中采用出租车的比例

则乘客需求量为：

$$N2 = n\sigma p$$

3） 当 N1 = N2，即供求平衡时，我们认为空驶率达到了理想数值，令两式相等：

$$(1-\mathrm{K})\mathrm{cN}\alpha = \mathrm{n}\sigma\mathrm{p}$$

可以求出：

$$K^* = 1 - \frac{n\sigma p}{N\alpha}$$

根据《2017 年北京交通发展年报》得知：北京市 2016 年出租车保有量为 68484 辆；出车率为 91%；日均接客人数约为 20 人次；北京市居民人口数为 2172.9 万人；居民平均每天出行次数为 1.22 次；居民出行工具选择中采用出租车的比例为 2.9%。将数据代入上式后求得空驶率理想值 K* = 0.38=38%

在空间角度上，由于获取的数据中，需求点和供应点的经纬坐标不对应，我们无法画出某一时刻空驶率的空间分布图，但我们以前面分别画出的 8 点时需求、供应分布图为例，可以看出，在高峰时段，市区总体供应量大于郊区，但是比例上看，相较于市区，郊区供求匹配度更好。

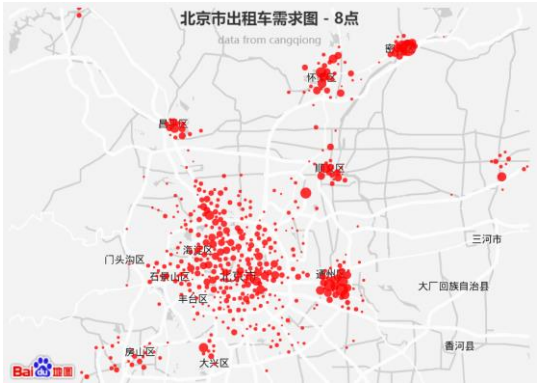


图 1-3 北京市 8 点出租车需求分布图

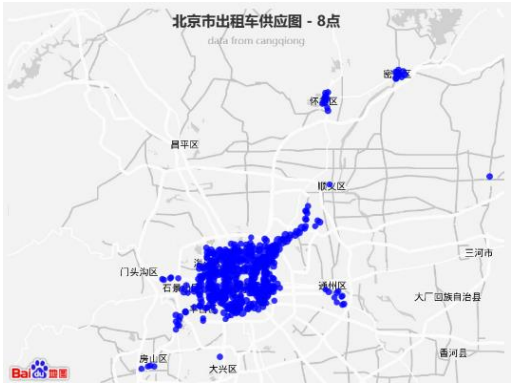


图 1-4 北京市 8 点出租车供应分布图

在时间角度上，我们画出了北京市 2016 年 8 月 6 日全天 24 小时每个时段空驶率的均值柱状图如下

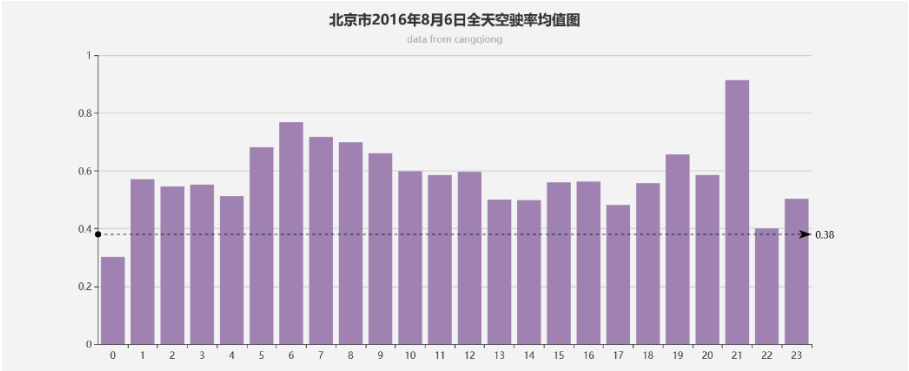


图 3 北京市 2016 年 8 月 6 日全天空驶率均值图

可以看出绝大多数时间，尤其是前面分析出的上下班高峰早上 7 点和下午 5 点左右，空驶率显著超过了理想值 38%，即空车比例较高，供过于求。这与我们的预判不同的原因可能是，出租车司机也都通过经验判断何时需求较多，选择相应时刻出车，但实际需求并不多，造成空驶较多；也可能是因为我们选取的数据是空间上包括郊区和市区的整个北京市、时间上 8 月 6 日是周六休息日。

2、人均响应车数

由前所述，人均相应车数的理想值η*=1，苍穹平台提供的数据中给出的是出租车和打车者在经度与纬度上的分布，我们可以利用这些数据算出实际的η。

假设以下参数：

Si —— 第 i 个行人的点

Di —— 第 i 个出租车的点

r —— 超参数，接客半径

对于某个叫车者，只有接客半径内的出租车可能会应答；同理，出租车只会接单接客半径内的叫车者。对每个行人维护一个应答数量 ni，对于每个出租车 Di，找到所有位于其接客半径内的行人，设共有 x 个人，则每个人的应答数量增加 1/ni。如下图所示，方块代表行人，星星代表出租车，以每个人为圆心，以出租车司机为了接单愿意行驶的最大距离为半径（意愿半径）画圆，如果出租车落在圆中则表示该出租车愿意接单，可以统计出每个人可以打到的出租车数量，A 的应答数量为 1.5，B 的为 0.5 。

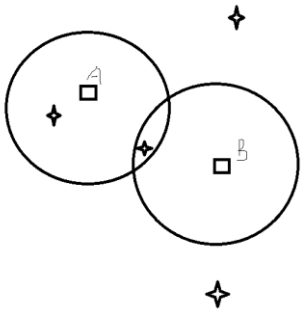


图 4 出租车和乘客位置示意图

易知，在供求完全平衡的情况下，所有行人的 ni 平均数应为 1 ，η=Σni /为行人总数，|η-1|即当前供求关系与标准供求关系的偏离程度。为了更好的衡量供求关系，将方差也考虑入内(否则 2 和 0 的平均数为 1，但并不平衡)。偏离程度修改为 X=|η-1|*(D(η)+1)。

编写程序对η值进行模拟，当接客半径取 0.01 时，各时段η值如下

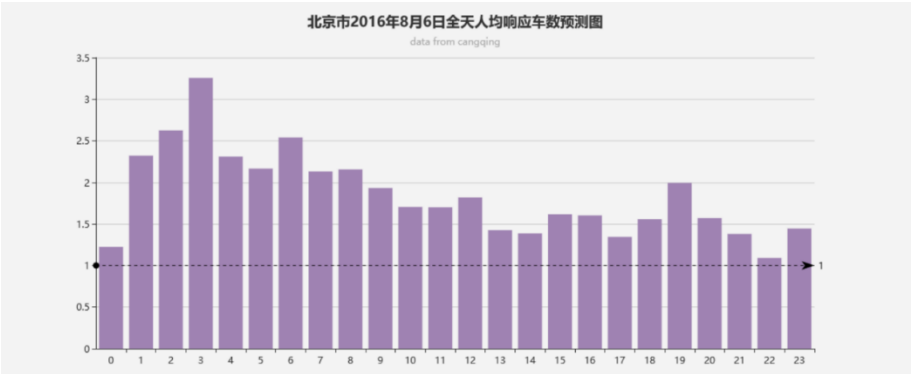


图 5 北京市 2016 年 8 月 6 日全天人均响应车数预测图

同时算出当日η均值为 1.85，方差为 0.255，偏离程度 X 为 1.06。可以看出人均响应车数大于理想值。

3、平均等待时间

首先确定平均等待时间的理想值

假设以下参数:

N —— 出租车数量

L —— 道路总长度

T —— 出租车平均运营时间

v —— 出租车平均车速度

α —— 平均出车率

K*——空驶率

假设一个人在一天 24 小时任意时间出门打车，此时运行在道路上空驶的车辆共有:

$$N\alpha\frac{T}{24}K^*$$

车辆的平均间隔为:

$$\frac{L}{N\alpha\frac{T}{24}K^*}$$

车辆扫过所有路面的时间为:

$$\frac{L}{N\alpha\frac{T}{24}K^*v}$$

因为乘客可能在道路的开头，也可能在结尾等待，所以等待时间的期望值为:

$$t=\frac{1}{2}\frac{L}{N\alpha\frac{T}{24}K^*v}$$

根据《2017 年北京交通发展年报》得知出租车数量为 68484 辆；道路总长度为 6373.5 公里；出租车平均运营时间为 12 小时*6 天/周=10.29h；出租车平均车速度为 31.35km/h；平均出车率约为 91%；空驶率理想值由前面求得为 38%。将数据代入上式得平均等待时间的理想值 t=36.04s。

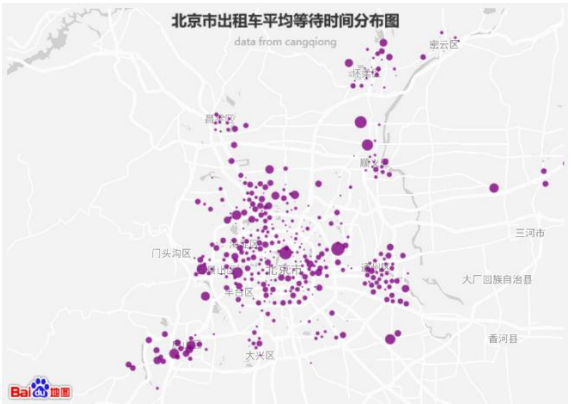


图 6 北京市 8 点出租车平均等待时间分布图

从空间上看，我们取了早上 8 点的分布图，此时所有点的平均等待时间平均值为 42.52s，由图中看出各地点等待时间较为均衡，个别点时间较长，市区乘客平均等待时间总体高于郊区乘客。

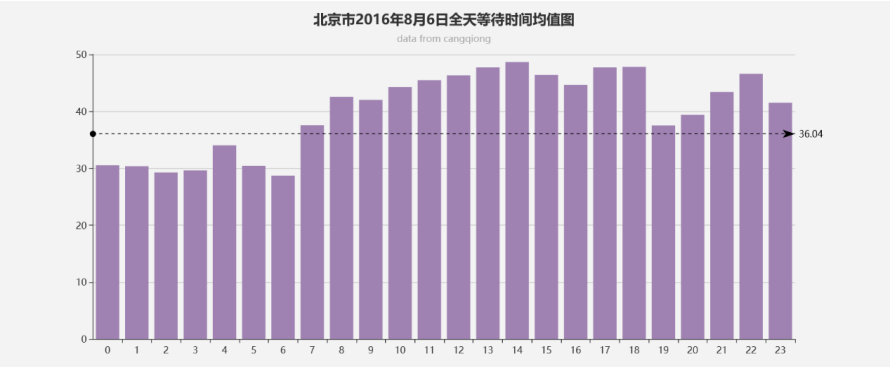


图 7 北京市 2016 年 8 月 6 日全天等待时间均值图

从时间上看，上图给出了北京市 2016 年 8 月 6 日全天各时段等待时间的平均值，可以看出实际等待时间多数超过理想值 36.04s。

5.1.4 结果分析

综合上述三个衡量指标，虽然空驶率普遍高于理想水平、人均响应车数大于理想值，供求较为平衡，但是平均等待时间较高，原因可能是司机积极性不高、接单前会进行比较而延缓时间，可能会影响乘客的满意度，打车难现象存在但不严重，后续可以通过补贴等政策提高接单率、降低平均等待时间。

5.2 问题二的建模与求解

5.2.1 指标确定

5.2.1.1 绘出补贴金额图像

问题二是分析出租车公司的补贴政策是否缓解了打车难的问题，我们首先搜集数据得到出租车公司的相关材料，这里我们选取滴滴打车和快的打车。考虑出租车公司的补贴政策，通过查阅相关资料，我们得到了滴滴打车和快的打车在不同时间段的补贴方案（表格中数据均为每单减免金额）。

时间	嘀嘀打车乘客补贴	快的打车乘客补贴
1 月 20 号	10	10
2 月 17 日	13	11
2 月 18 日	16	13
3 月 4 日	10	10
3 月 5 日	4	5
3 月 22 日	0	4
5 月 17 日	0	0
7 月 9 日	0	0
8 月 9 日	0	0

表 1-1 2016 年 1-8 月滴滴打车和快的打车对乘客补贴额度对比

时间	嘀嘀打车司机补贴	快的打车司机补贴
1 月 20 号	10	10
2 月 17 日	10	8
2 月 18 日	10	10
3 月 4 日	10	10
3 月 5 日	8	6
3 月 22 日	6	6
5 月 17 日	2	6
7 月 9 日	0	2
8 月 9 日	0	0

表 1-2 2016 年 1-8 月滴滴打车和快的打车对司机补贴额度对比

我们以时间 t 为横坐标，对乘客和司机的补贴金额λ_1，λ_2为纵坐标，用绘图软件 EChart 绘出不同时间两家公司的补贴金额折线图，如下图所示：

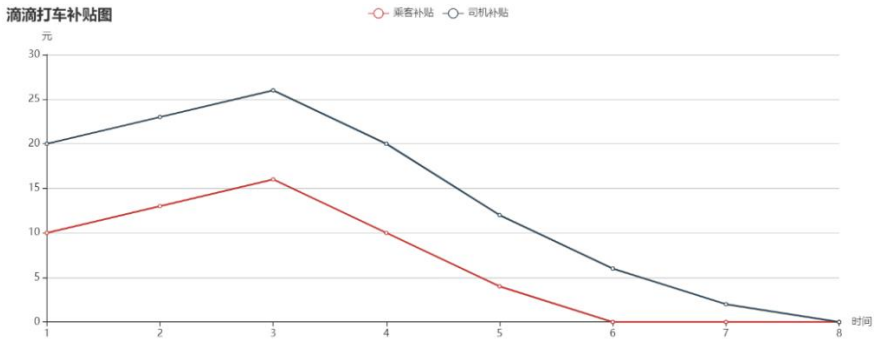


图 8 滴滴打车补贴折线图

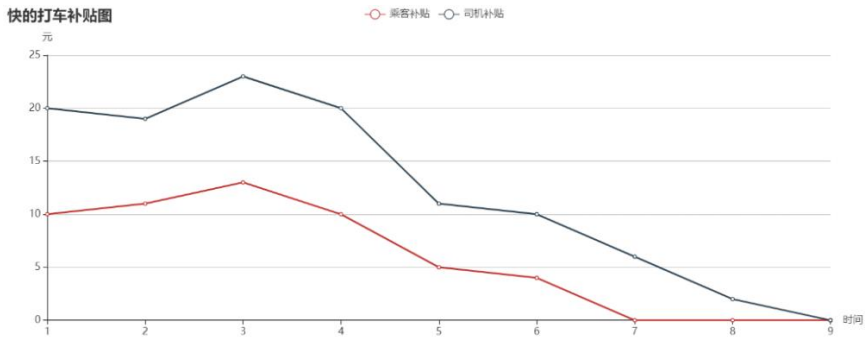


图 9 快的打车补贴折线图

由于是研究补贴的影响，所以我们去除没有补贴的数据，由上图我们可以求出滴滴打车对乘客的平均补贴金额λ_1是 10.6 元，对司机的平均补贴金额λ_2为 8 元。快的打车对乘客的平均补贴金额λ_1是 8.8 元，对司机的平均补贴金额λ_2为 7.3 元。

5.2.1.2 确定居民出行工具选择中采用出租车的比例 p

对乘客和司机补贴，不会增加人口总数和出租车总数，但是显然会增加居民出行工具选择中采用出租车的比例 p，因为可能存在由于经济原因本来不想打车，但是有补贴后认为经

济负担可以承受，选择打车的情况。简单来说，随着对乘客的补贴的增加，居民出行工具选择中采用出租车的比例 p 会增多。然后对这个关系建模，对 p 来说，在没有补贴时它有初始值a_1，随着补贴增大，它们逐渐趋于 1。可以用指数函数来模拟。对乘客的补贴为λ_1，公式为：

$$\mathrm{p} = 1 - a_1 * e^{-k_1\lambda_1}$$

需要注意的是，没有补贴时，p=1-a_1，a_1可以通过查阅资料获得。

5.2.1.3 确定出租车的出车率α

同样的，对乘客和司机的补贴，不会增加人口总数和出租车总数，但是显然会增加出租车的出车率α，因为对于司机进行补贴后，会给司机更多的动力，更希望出车，这样可以更快的赚到以前可以赚到的钱。简单来说，随着对司机的补贴的增加，出租车的出车率α会增多。然后对这个关系建模，对α来说，在没有补贴时它应该有个初始值a_2，随着补贴增大，它们逐渐趋于 1。可以用指数函数来模拟。对司机的补贴为λ_2，公式为：

$$\alpha = 1 - a_2 * e^{-k_2\lambda_2}$$

需要注意的是，没有补贴时，α=1-a_2，a_2可以查阅资料获得。

最后解释居民出行工具选择中采用出租车的比例 p 和出租车的出车率α公式中的参数k_1，k_2的选取。对于k_1和k_2，从逻辑分析上来说，补贴对乘客的影响要更大，因为出租车司机本身的职业因素，他们无论有没有补贴都会更大的几率出车，所以补贴只会小幅度的影响出租车的出车率。举个例子，可能区别更多的是没有补贴工作 8 小时，有补贴工作 9 小时。而乘客的浮动性较大，乘客有很多交通工具可以选择，所以补贴对他们的影响更大，所以k_1应该大于k_2。k_1和k_2的具体选择会在具体实施环节中详细说明。这样，通过这两个问题的分析和建模，总可用出租车数和总乘客数就可以看作λ_1，λ_2的函数。

5.2.1.4 确定接客半径 R

对于接客半径 R，可将其视作补贴金额的线性函数：$R=R_0+k*\lambda_2$。R_0是没有补贴时的接客半径。λ_2是司机补贴。k 是对于司机补贴λ_2变量改变的控制参数。

对于求 k 的值，我们将使用出租车司机行驶单位路程的成本 c。出租车司机行驶单位路程的成本包括燃油成本 a 和机会成本 b。燃油成本查阅资料可知，每公里的燃油成本大概在 0.35-0.75 元左右。机会成本需要我们进行计算，先查阅出租车司机的日均收入，再查阅出租车司机的日均行驶里程，两者做商即得到他们行驶单位里程的机会成本。2016 年北京出租车司机的日均收入大致是 117 元，出租车司机的日均行驶里程为 450 公里。由于成本越高，接客半径越小，所以得到 k 的公式为 k =1/c。

5.2.1.5 确定打车缓解程度指标s

对于分析出租车公司的补贴方案是否缓解打车难，我们先研究第一个问题中的两个参数：等待时间 t 和人均可应答车辆数η两个指标。首先是 t，由于第一问已经清楚的介绍过 t 的公式，这里直接给出公式。公式如下：

$$t = \frac{1}{2}\frac{L}{N\alpha\frac{T}{24}K^*v}$$

而第一问中也详细说明了K^*的公式，如下

$$K^* = 1 - \frac{n\sigma p}{N\alpha c}$$

将 K*带入 t 可以得到：

$$t = \frac{1}{2}\frac{L}{\frac{T}{24}v(N\alpha c - n\sigma p)}$$

再将出租车的出车率α和居民出行工具选择中采用出租车的比例 p 的关系式带入，即可求出 t 的变化：

$$t = \frac{12L}{Tv(Nc - n\sigma - Nca_2e^{-k_2\lambda_2} + n\sigma a_1e^{-k_1\lambda_1})}$$

得到 t 随λ_1和λ_2变量变化而变化的公式，其他的变量都是问题一中求得的常量。

对于指标人均可应答车辆数η，我们在第一问中已经求解出了它的公式，给出了用于计算的相对应的代码。计算人均可应答车辆数η时我们需要变量接客半径 R，由于我们可将其视作补贴金额的线性函数：$R=R_0+k*\lambda_2$，所以人均可应答车辆数η随着司机补贴λ_2变化而变化。对于计算时需要需求信息和分布信息，我们可以随机生成数据点来模拟。在一定区域内，出租车和乘客的分布可近似视作二维正态分布，分布的均值对应现实中的市中心。可按照上文中的公式，根据λ_1和λ_2给出出租车和乘客的总点数，以及接客半径 R，按照问题 1 中的方法计算η。在模拟时，我们会将正态分布的方差取大一些。

得到可以表示打车难度的指标等待时间 t 和人均可应答车辆数η后，需要综合这两个指标得到打车缓解程度指标s的表达式。对于量化对打车难的缓解程度，我们先设没有补贴的时候，补贴λ_1和λ_2为 0，算出此时的t^*和η^*，然后再将补贴带入，算出此时的 t 和η。如果打车难度得到缓解，那么 t 应该减少，或者η应该增大，将会导致 s 变大，也就是说明打车难度得到缓解。我们使用下面的公式来量化缓解程度：

$$s = \lg\frac{t^*}{t} + lg\frac{\eta}{\eta^*}$$

需要注意的是，如果没有缓解打车难问题，s 会取到负值。

5.2.2 具体实现/模型求解及结果分析

5.2.2.1 使用模型求出居民出行工具选择中采用出租车的比例 p

由于通过之前查阅的资料，我们知道没有补贴时的 p 是 2.9%，所以a_1是 1-0.029。在这里我们对于k_1取值为 0.02 更合适，对乘客的补贴变量λ_1变化的区间为 0 到 30 元，步长为 1。使用 python 画出相对应的图，得到 p 与λ_1对应变化的直观理解。

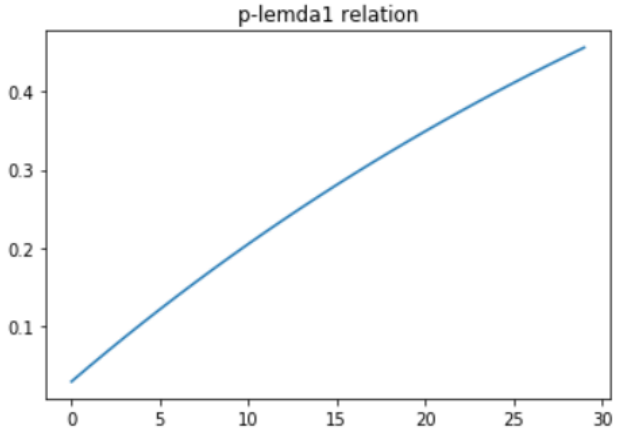


图 10 采用出租车比例与乘客补贴曲线图

显然，随着λ_1的增加，p 也会增加，且λ_1达到一定值后会逐渐放缓增长速度。

5.2.2.2 通过模型获得λ_2和出租车的出车率α的关系

2016 年北京出租车司机的日均收入大致是 117 元，出租车司机的日均行驶里程为 450 公里。经过计算得到k_2为 0.01。由于通过之前查阅的资料。我们知道没有补贴时的α是 91%，所以α_1是 1-0.91，对司机的补贴变量λ_2变化的区间为 0 到 30 元，步长为 1。

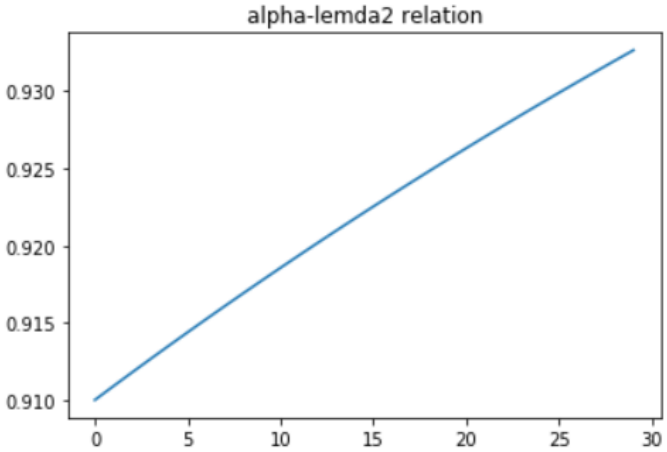
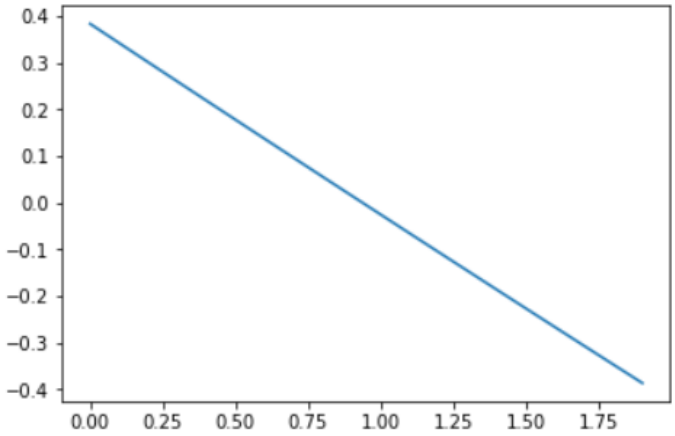


图 11 出租车的出车率与司机补贴曲线图

5.2.2.3 通过模型求解 K

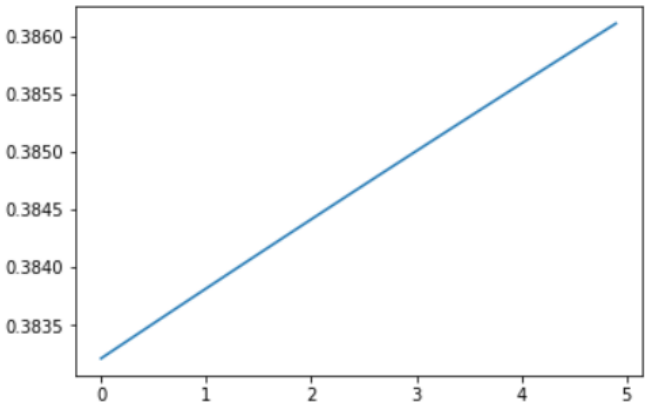
首先根据公式和查阅的材料，我们绘制了空驶率 K 关于λ_1的函数。



‘图 12 空驶率与乘客补贴曲线图

上图可以看出，K 随着λ_1的增大迅速减小，减小的速度与k_1的大小有关，可以发现一个现象，K 竟然慢慢小于 0 了.这是因为如果λ_1太大，那么打车的人会过多，以至于就算所有的出租车上都载着人，也满足不了乘客需求。导致空驶率出现负值。

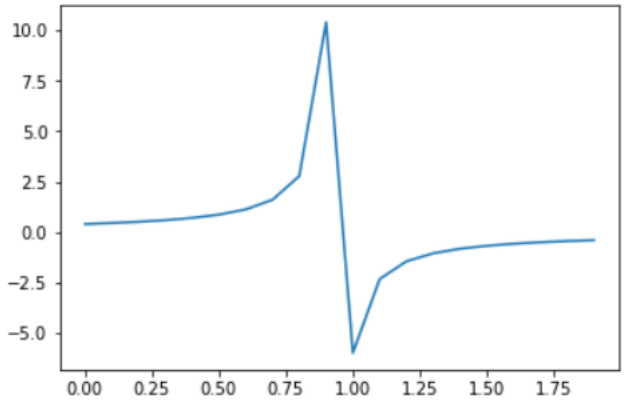
绘制空驶率 K 关于λ_2的函数:



‘图 13 空驶率与司机补贴曲线图

5.2.2.4 通过模型求解 t

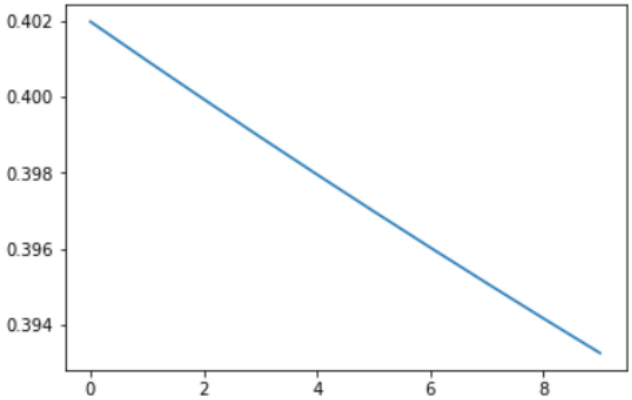
根据公式和查阅的材料，绘制 t 关于λ_1的函数，假设λ_1= 0。



‘图 14 等待时间与乘客补贴曲线图

上图可以看出平均等待时间一开始确实变大了，之后变小是因为 K 变成负了，这里的数据也就没有意义了。

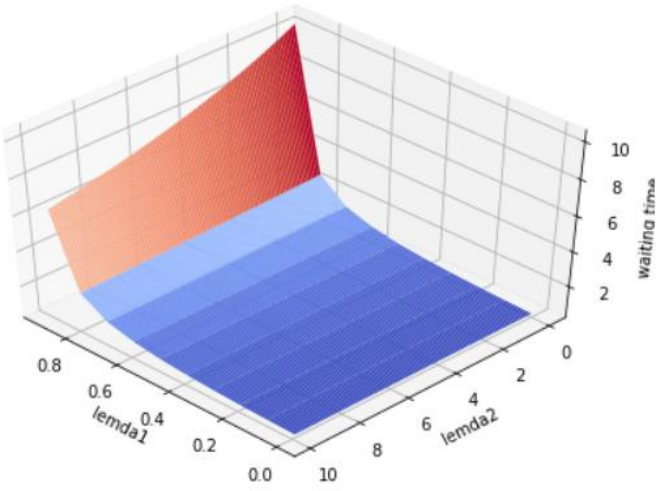
根据公式和查阅的材料，绘制 t 关于λ_2的函数，假设λ_1= 0。



‘图 15 等待时间与司机补贴曲线图

5.2.2.5 t 和λ_1与λ_2的函数关系

根据公式和查阅的材料，通过 t 和λ_1与λ_2同时变化时的函数绘制关系图。

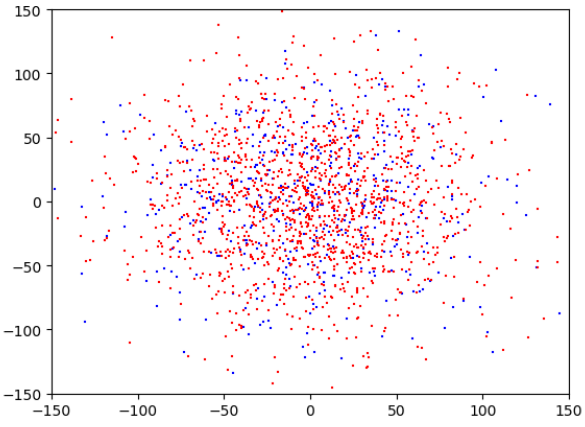


‘图 16 等待时间与补贴曲线图

通过这张图我们可以直观的感受到，λ_1或者λ_2增长会使得 t 增长，并且λ_1对 t 的影响更大。首先我们已经知道λ_1和λ_2都是 0 的时候，t 是 24s。带入滴滴打车和快的打车的数据，其中，滴滴的补贴数据是对乘客的平均补贴金额λ_1是 10.6 元，对司机的平均补贴金额λ_2为 8 元。带入公式求得 t 为 19.9s。快的的补贴数据是对乘客的平均补贴金额λ_1是 8.8 元，对司机的平均补贴金额λ_2为 4.3 元。带入公式求得 t 为 19.2s。

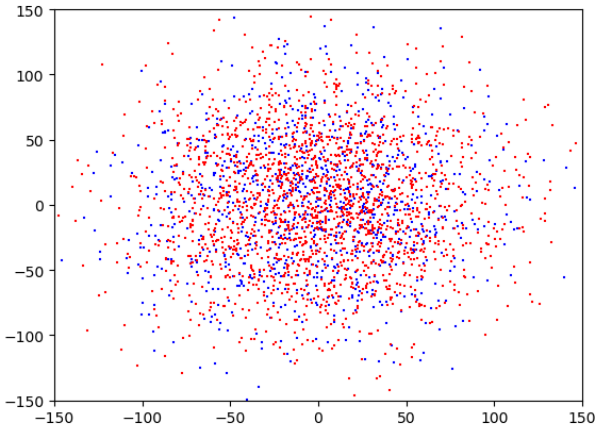
5.2.2.6 指标人均可应答车辆数η

先编写代码，求得了模拟的需求和供应数据，使用了随机函数。运行函数，可以得出模拟的符合正态分布的数据点图。



‘图 17 模拟需求供应图 1

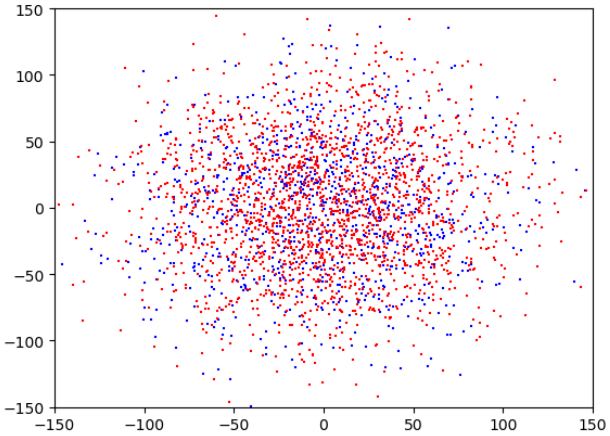
通过设置参数λ_1和λ_2都为 0，表示没有补贴，运行代码，得到人均可应答车辆数η为 0.96。我们使用滴滴的补贴数据，对乘客的平均补贴金额λ_1是 10.6 元，对司机的平均补贴金额λ_2为 8 元。将数据带入后，可得模拟的需求供应图变为：



‘图 18 模拟需求供应图 2

可以看出供应和需求量都明显增多。同时得到人均可应答车辆数η成为了 1.53。比之前有所增加。说明滴滴的打补贴策略是缓解了打车难的人均可应答车辆数这一指标。

之后，我们使用快的的补贴数据，对乘客的平均补贴金额λ_1是 8.8 元，对司机的平均补贴金额λ_2为 7.3 元。将数据带入可得模拟的需求供应图变为：



‘图 19 模拟需求供应图 3

可以看出供应和需求量都明显增多。同时得到人均可应答车辆数η成为了 1.50。比滴滴有略微的减少。说明快的的补贴策略是缓解了打车难的人均可应答车辆数这一指标。

5.2.2.7 打车难缓解程度 s

根据前面得到的滴滴和快的补贴策略下，等待时间 t 和人均可应答车辆数η的值。现在可以根据之前的公式，求出缓解程度，缓解程度计算公式为：

$$s = \lg\frac{t^*}{t} + lg\frac{\eta}{\eta^*}$$

所以滴滴的缓解程度 s 为 0.195；快的的缓解程度 s 为 0.669。

5.2.3 结果分析

通过观察模型的求解结果，我们得到以下分析：

1）打车补贴推广前后比较：由上图可以看出，两个公司的缓解程度分布范围在 0.1 到 0.7 之间，说明滴滴打车和快的打车两个公司的投对乘客打车难的问题是有一定缓解的，但这个缓解效果并不是很大。据新闻数据显示，滴滴快的两个公司对补贴的投入总金额甚至达到了 19 亿，可谓是一个烧钱的补贴。事实上，两个公司之所以要进行补贴的根本目的并不完全是要缓解打车难问题，主要还是因为两个公司为了抢占客户量，只是这样的竞争战顺带对打车难问题有了一定的缓解。

综上，两个公司的补贴方案确实是对打车难问题有一定缓解，但是缓解程度并不理想。

2）两公司之间分析：由数据可知，快的打车对于打车难问题的缓解程度更大。

3）综合分析，可以看出，两个打车公司的补贴方案带来了一定程度的缓解。单从缓解打车难问题看，这种补贴方案缺乏一定的针对性。针对这个问题，我们给出了第三问可以平衡公司利润和打车难的优化补贴模型。

5.3 问题三的建模与求解

问题二中两家公司的补贴方案虽然对打车难问题有一定的缓解，但效果并不明显，并且补贴方案仅考虑了供需平衡，而为考虑公司利润问题。接下来我们尝试设计一个合理的补贴方案满足下面两个条件：①从乘客的角度出发，改善打车难的问题；②从打车软件平台的角度出发，使利润最大化。

5.3.1 补贴模型的建立

1、利润函数

直观来看，利润跟接单数量和每单金额有关

1）接单数量 m

从乘客的角度出发，考虑下面几个大家都已经熟悉的指标

n —— 居民人数

σ —— 居民平均每天出行次数

p —— 居民出行工具选择中采用出租车的比例

δ —— 选择出租车的时候使用该打车软件的比例

则接单数量为：

$$m = n\sigma p\delta$$

在问题二中已知 p 是关于补贴方案λ_2的函数，接下来就是确定 δ，我们认为δ是关于乘客满意度的函数，设乘客满意度为 f，则打车时选择该软件的比例

$$\delta = 1 - e^{kf}$$

下面就是确定 f，对于满意度，很容易可以想到满意度和等待时间 t 有关，等的时间越短，满意度越高。所以 f 是关于 t 的函数

$$f = func(t)$$

在这里，我们利用前面从滴滴平台获取的实际数据拟合一个浅层神经网络，用于拟合 f 与 t 的关系，这个关系就记为 func，表达式的形式为 y= sigmoid(linear2(tanh(linear1(x)))，其中两个线性层的参数为：

```
In [164]: model.l1.weight
Out[164]: Parameter containing:
          tensor([[0.4241],
                  [0.6125],
                  [0.7225],
                  [0.1596],
                  [0.1607]], requires_grad=True)

In [165]: model.l2.weight
Out[165]: Parameter containing:
          tensor([[ 0.1602,  0.1650,  0.3963, -0.5008, -0.8633]], requires_grad=True)
```

拟合结果如下图，符合我们的预测客满意度随着等待时间增大而不断下降，且下降速度逐渐变缓。

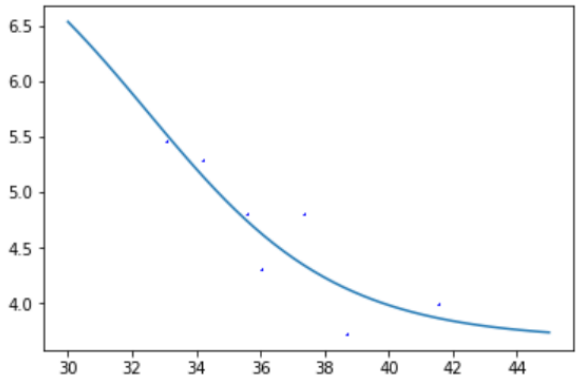


图 20 满意度与等待时间关系图

由于在问题二中已知 t 和λ_1、λ_2有关。所以 m 也和λ_1、λ_2有关。

2）每单收取的钱：将我们获得的订单价格求平均值，得毛利β为 95.33 元

F2 =AVERAGE(E2:E5920)

	A	B	C	D	E	F
1		hour	longitude	latitude	money	average
2	0	0	116.3461	39.9987	263	95.33468
3	1	0	116.3165	39.8661	24	
4	2	0	116.5218	39.8317	35	
5	3	0	116.4293	40.0408	81	
6	4	0	116.2257	39.9117	80	
7	5	0	116.4225	39.9569	158	
8	6	0	116.3699	39.778	16	
9	7	0	116.3165	40.0586	498	

图 21 订单金额统计表

3）利润函数简单定义为单数乘上每单平台净利润，即每单金额减去对司机和乘客的补贴，综合 1）和 2），可得利润函数：

$$\gamma = m * (\beta - \lambda_1 - \lambda_2) = n\sigma p\delta * (\beta - \lambda_1 - \lambda_2)$$

相当于我们的λ_2可以提高整体的 p，但是其实因为我们使一部分人使用我们的打车软件，所以打车的多了。在问题二中我们并不关心这里面有多少人是使用我们的软件的，但是在问题三中我们必须弄清楚，所以才引入了δ，用来刻画这个比例。

2、约束条件

如果单纯追求一时的高利润，而不考虑顾客满意程度也会逐渐失去用户，所以在赚取较多利润的同时，要保证打车难问题受到改善，即λ_1，λ_2并不是任意取的，要满足问题二中的 s>0:

$$s = \lg\frac{t^*}{t} + lg\frac{\eta}{\eta^*}$$

5.3.2 补贴模型的求解

接下来的问题就简单了，设定一个λ_1，λ_2的范围，比如 0~100，选定一个步长，如 0.1，所以（λ_1，λ_2）一共有 1000*1000 个点。然后每个点利用利润公式可以算出利润，同时用 s 来判断这样一个点是不是合法的，就可以找出最优的λ_1，λ_2。

六、模型的评价

6.1 模型的优点

（1）将出租车资源“供求匹配”程度衡量问题转换为空车率、人均响应车数、平均等待时间三个指标的求解问题，其中以人为圆心、司机接客半径画圆，记录覆盖范围内出租车数量的人均响应车数指标的选定较为新颖，不同于万人拥有量等传统指标；创新还体现在一些结合查阅资料和调查体验自定义的公式上，综合各方面参数量化打车难缓解程度、净利润等指标。

（2）在模型求解时，运用模拟的方式进行数据采集，得到了具体数据结果，有较强说服力，较好的解决了数据缺乏的问题；对数据进行了很好的可视化工作，采用地图散点图、折线图、柱状图、拟合曲线、模拟曲面等图形，直观表现出指标量值大小、时间空间分布以及变化趋势，便于统计和定性分析。

（3）三个模型环环相扣，相互辅佐论证。

6.2 模型的缺点

（1）数据来源仅为滴滴快的，尽管两者占据较大市场份额，但实际情况仍需要考虑其它市场份额的竞争问题。

（2）由于真实数据难以搜集全面，数据缺乏的问题使得我们无法对模型进行强有力的支撑与验证，只能通过程序的模拟间接处理。

6.3 模型的改进

（1）在对供需平衡做出评价时，区分平时与周六日、节假日会使结果更为准确。

（2）对于补贴，我们只求出了一个单一的值，如果可以结合空间复杂度和时间复杂度来制定一个动态的补贴方案，比如根据高峰期和低谷期，不同区域的特征形成不同的补贴方案。这样可能会对缓解打车难有更多的帮助。

（3）可以按照不同人群确立不同的补贴方案，例如为鼓励老年人使用智能手机、使用打车软件，适当提高补偿标准。

（4）横向对比更多城市打车难问题现状和解决办法。

6.4 模型的推广

本文通过建立供求匹配模型、缓解程度判断模型和利润最大化模型，对北京市出租车资源的供求匹配程度和现有补贴方案进行了分析评价，并设计了合理的补贴方案。所建模型具有较好的实用性，对于今后实际出租车资源配置问题解决具有一定参考价值。

此外，本课题制定的评估规则不只局限与打车的供求关系，也可推广至其它供求关系的衡量中，例如“外卖接单”时空分布的供求问题中，从而在创造利润的同时最大化供求双方的满意程度，具有很强的现实意义。

七、参考文献

[1] 姜启源，谢金星，数学模型（第五版）[M]. 北京：高等教育出版社，2018.
[2] 苍穹. 滴滴快的智能出行平台[DB/OL]. 2015-09-11, [2015-09-12]. http://v.kuaidadi.com/
[3] 北京交通发展研究院，2017 年北京交通发展年报，http://www.bjtrc.org.cn/List/index/cid/7.html, 2019-5-10.
[4] 北京市统计局，北京统计年鉴-2017，http://tjj.beijing.gov.cn/nj/main/2017-tjnj/zk/indexch.htm, 2019-5-10.
[5] 中华人民共和国中央人民政府，北京市第五次综合交通调查结果出炉，http://www.gov.cn/xinwen/2016-07/07/content_5089031.htm, 2019-5-10.
[6] 曹祎，罗霞. 基于打车软件使用率的出租车运营速度分析[J] . 重庆交通大学学报自然科学版,2015,34(6):123-127.
[7] 刘嘉琪，邹泞憾，周梓楠，王颖酷. 互联网时代出租车供需匹配及补贴方案确定[J]. 经济数学,2016,(02):103-110.
[8] 李冬新，栾洁，滴滴打车的营销策略与发展对策研究[N]，青岛科技大学学报(社会科学版), 31(1): 2015.
[9] 张文全，影响城市出租车供求关系的因素分析，河北交通职业技术学院学报第 8 卷第 1 期: 1-3 页, 2011.

附录

附录 A. 问题一中计算人均响应车数 η 的代码

```
import numpy as np
import pandas as pd

demand = pd.read_csv("demand_2016.08.06_110100_.csv")
distribute = pd.read_csv("distribute_2016.08.06_110100_.csv")

r = 0.01

#统计 0 点到 23 点
demand_data = []
distribute_data = []
myindex = []
times = [i for i in range(24)]
result = [0 for _ in range(len(times))]

for i in range(len(times)):
    index1 = (demand.hour == times[i])
    index2 = (distribute.hour == times[i])
    demand_data.append(demand[index1])
    distribute_data.append(distribute[index2])
    index = [0 for _ in range(len(demand[index1]))]
    myindex.append(index)

for i in range(len(times)):
    cars = distribute_data[i]
    men = demand_data[i]
    cars_number = cars.value.values
    men_number = men.value.values
    cars_longitude = cars.longitude.values
    men_longitude = men.longitude.values
    cars_latitude = cars.latitude.values
    men_latitude = men.latitude.values

    number1 = len(cars_number)
    number2 = len(men_number)
    for n in range(number1):
        alltargetmen = []
        mennumber = 0
        carnumber = cars_number[n]
        for m in range(number2):
```

```
            if (cars_longitude[n]-men_longitude[m])*(cars_longitude[n]-men_longitude[m])+(cars_latitude[n]-men_latitude[m])*(cars_latitude[n]-men_latitude[m]) <= r*r:
                alltargetmen.append(m)
                mennumber = mennumber + men_number[m]
        if mennumber == 0:
            continue
        added = carnumber/mennumber
        for target in alltargetmen:
            myindex[i][target] = myindex[i][target] + added


    number = len(myindex[i])
    allthecars = sum(cars_number)
    overall = 0
    for j in range(number):
        overall = myindex[i][j] * men_number[j] + overall
    allmen = sum(men_number)

    result[i] = overall / allmen

mean = np.mean(result)
var = np.var(result)
X = abs(mean - 1) * (var + 1)
#偏离程度 X=|η-1|*（D(η)+1）

print(result)
print(mean, var, X)
```

附录 B. 问题二中图像绘制代码

a. 选择出租车出行概率 p - 乘客补贴λ1 关系图

```
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
#define p, k1 需要再确定
a1 = 1 - 0.029
k1 = 0.02

def P(lemda1):
    return 1 - a1*(np.e**(-k1*lemda1))

print(P(0))
lemdas = np.arange(0, 30, 1)
ps = [P(i) for i in lemdas]
```

```
plt.title("p-lemda1 relation")
plt.plot(lemdas, ps)
```

b. 出租车出车率α – 司机补贴λ2 关系图

```
#define alpha, k2 需要再确定
a2 = 0.09
k2 = 0.01

def Alpha(lemda2):
    return 1 - a2*(np.e**(-k2*lemda2))

print(Alpha(0))
lemdas = np.arange(0, 30, 1)
ps = Alpha(lemdas)
plt.title("alpha-lemda2 relation")
plt.plot(lemdas, ps)
```

c. 空驶率 K、平均等待时间 t 与乘客补贴 λ1、司机补贴 λ2 的关系图

```
#define K
N = 68484
n = 21729000
c = 20
sigma = 1.22

def K(p, alpha):
    return 1 - (n*sigma*p)/(N*alpha*c)

print(K(P(0), Alpha(0)))
#define t 下面这些参数需要查找资料确定
T = 10
L = 80000
v = 10

def t(lemda1, lemda2):
    p = P(lemda1)
    alpha = Alpha(lemda2)
    k = K(p, alpha)
    numerator = 12*L #12*L*c
    denominator = N*alpha*T*k*v #T*v*(N*alpha*c - n*sigma*p)
    #print(denominator)
    return numerator/denominator

print(t(0, 0))
```

```
#绘制空驶率 K 关于 lemda1 的函数
lemda1 = np.arange(0, 2, 0.1)
k = K(P(lemda1), Alpha(0))
plt.plot(lemda1, k)
#绘制空驶率 K 关于 lemda2 的函数
lemda2 = np.arange(0, 5, 0.1)
k = K(P(0), Alpha(lemda2))
plt.plot(lemda2, k)
#绘制 t 关于 lemda1 的函数，假设 lemda2 = 0
lemda1 = np.arange(0, 2, 0.1)
time = t(lemda1, 0)
plt.plot(lemda1, time)
#绘制 t 关于 lemda2 的函数，假设 lemda1 = 0
lemda2 = np.arange(0, 10, 1)
time = t(0, lemda2)
plt.plot(lemda2, time)
# t(0,0)表示的是没有补贴时候的 t
# t(lemda1, lemda2)表示有补贴时候的 t
lemda1 = np.arange(0, 1, 0.1) #这个取值范围里可以保证 K 非负
lemda2 = np.arange(0, 10, 0.1)
lemda1, lemda2 = np.meshgrid(lemda1, lemda2)
time = t(lemda1, lemda2)
fig = plt.figure()
ax = Axes3D(fig)
#ax.contour3D(lemda1, lemda2, time)
ax.plot_surface(lemda1, lemda2, time, rstride=1, cstride=1, cmap=plt.cm.cool
warm)

ax.set_xlabel('lemda1')
ax.set_ylabel('lemda2')
ax.set_zlabel('waiting time')

ax.view_init(45, 135) #第一个参数是仰视角度，第二个是侧视角度

plt.show()
```

d. 模拟人车分布计算η散点图

```
import numpy as np
from numpy.linalg import cholesky
import matplotlib.pyplot as plt
np.random.seed(0)
#以下超参是可以更改的
k1=1
k2=0.5
```

```
cost=5 #出租车司机行驶单位里程所需要的成本
r0=3 #没有补贴时的接客半径
lambda_1=0
lambda_2=0  #司机补贴
a1=0.5
a2=0.3
sampleNo_init_car = 2000 #测试时车的数据点数上限
sampleNo_init_man = 1000 #测试时人的数据点数上限
test_num=10 #测试 10 次取平均
mu = 0 #正态分布的均值和方差
sigma = 50

#以下参数是算出来的
sampleNo_man=int((1-a1*(np.e**(-k1*lambda_1)))*sampleNo_init_man)
sampleNo_car=int((1-a2*(np.e**(-k2*lambda_2)))*sampleNo_init_car)
r=r0+lambda_2/cost  #接客半径

result=[]
for test_idx in range(test_num):
    #生成数据点
    man_x=np.random.normal(mu, sigma, sampleNo_man)
    man_y=np.random.normal(mu, sigma, sampleNo_man)
    car_x=np.random.normal(mu, sigma, sampleNo_car)
    car_y=np.random.normal(mu, sigma, sampleNo_car)
    myindex=[0 for _ in range(sampleNo_man)]

    for i in range(sampleNo_car):
        alltargetman=[]
        for j in range(sampleNo_man):
            if (man_x[j]-car_x[i])**2+(man_y[j]-car_y[i])**2<=r**2:
                alltargetman.append(j)
        if len(alltargetman)==0:
            continue
        added=1/len(alltargetman)
        for target in alltargetman:
            myindex[target]+=added

    overall=sum(myindex)
    result.append(overall/sampleNo_man)
print('average yita:',sum(result)/test_num)

#画图
plt.scatter(man_x,man_y,s=1,marker='+',c=(0,0,1))
plt.scatter(car_x,car_y,s=1,marker='+',c=(1,0,0))
```

```
plt.xlim((-150, 150))
plt.ylim((-150, 150))
plt.show()
```

附录 C. 问题三中拟合顾客满意度 f 与平均等待时间 t 函数神经网络代码

```
import numpy as np
import pandas as pd
import os
#读取北京市的数据
times = [0,8,13,20]
response=[]
satisfy=[]
citys=os.listdir('data')
for city in citys:
    files=os.listdir('data/'+city)
    for file in files:
        f=pd.read_csv('data/'+city+'/'+file)
        if file.startswith('response'):
            response.append(sum(list(f.value))/len(list(f.value)))
        elif file.startswith('satisfy'):
            satisfy.append(sum(list(f.value))/len(list(f.value)))

break

data=[]
for i in range(len(response)):
data.append([response[i],satisfy[i]])

#训练网络
import torch
import torch.autograd as autograd
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torch.autograd import Variable

class MyDataset(Dataset):
    def __init__(self,dataset):
        self.full_batch=dataset

    def __len__(self):
        return len(self.full_batch)

```

```
    def __getitem__(self,index):
        response=self.full_batch[index][0]
        satisfy=self.full_batch[index][1]
        return  response,satisfy

class Model(nn.Module):
    def __init__(self, num):
        super(Model, self).__init__()
        self.l1=nn.Linear(1,num)
        self.l2=nn.Linear(num,1)
    def forward(self,x):
        x=self.l1(x)
        x=F.tanh(x)
        x=self.l2(x)
        x=F.sigmoid(x)
        return x

dataset=MyDataset(data)
model=Model(5)
dataloader=DataLoader(dataset,shuffle=True,batch_size=7)
optimizer=optim.Adam(model.parameters(),lr=0.001)

loss_seq=[]
for i in range(10000):
    for response,satisfy in dataloader:
        optimizer.zero_grad()
        response=response.resize(7,1).float()
        satisfy=satisfy.float()/10
        y=model(response)
        y=y.squeeze()
        loss=torch.mean((y-satisfy).pow(2))
        loss.backward()
        optimizer.step()
        loss_seq.append(float(loss))

#输出拟合结果
#拟合结果
x=np.linspace(30,45,50)
y=model(torch.from_numpy(x).float().resize(50,1))
y=y.detach().numpy()
plt.scatter(response,satisfy,s=1,marker='+',c=(0,0,1))
plt.plot(x,10*y)
plt.show()
```