一、问题描述
变量定义：plc_num：地点总数，people：候选总人数，candidate_num：最终选择人数，k_favor：最常去的地点数

将地点序列化为一个长为plc_num的比特串B，每个用户在B中选择k_favor个最常去的地点标记为1，其余地点为0。服务器收到的比特串S经过RAPPOR算法处理。目标为通过S在people中选取人数为candidate_num的群体，使得他们的真实覆盖率最大。

二、算法描述
对于每个地点l_u，记此地在真实记录中的访问人数为随机变量A，若没有一个人真实访问，即对应A=0。对选取的所有志愿者的S，第u个比特处的求和记为X_u。根据RAPPOR机制，可以得到后验概率：(N=candidate_num)

$$P(A=0\,|\,X_u)=\frac{P(A=0,X_u)}{\sum_{i=0}^{N}P(A=i)P(X_u\,|\,A=i)}$$
$$=\frac{C_N^{X_u}p_{01}^{X_u}(1-p_{01})^{N-X_u}(1-\xi)^N}{\sum_{i=0}^{N}C_N^i\xi^i(1-\xi)^{N-i}\sum_{m=max(0,X_u+i-N)}^{min(i,X_u)}C_i^m p_{11}^m(1-p_{11})^{i-m}C_{N-i}^{X_u-m}p_{01}^{X_u-m}(1-p_{01})^{N-i-X_u+m}}$$

（推导过程：
核心在于计算$P(A=i)P(X_u\,|\,A=i)$。
$P(A=i)$即N枚硬币中有i枚正面朝上的先验概率，是一个二项分布，即$C_N^i\xi^i(1-\xi)^{N-i}$。
$P(X_u\,|\,A=i)$表示在原先有i枚正面朝上的硬币的条件下，翻转后变为X_u枚的概率。考虑这X_u枚正面硬币的来源，可以分为两类：1、原先正面朝上的硬币，翻转后仍为正面朝上，转移概率为p_{11}；2、原先反面朝上的硬币翻转后成为正面朝上，转移概率为p_{01}。1中假设有m枚硬币原先正面朝上，翻转后保持不变，那么$i-m$枚硬币由原先的正面向上变为反面向上，这是一个二项分布，概率为$C_i^m p_{11}^m(1-p_{11})^{i-m}$。此时2中应当有$X_u-m$枚硬币由反面向上翻转为正面向上，$N-i-X_u+m$枚硬币仍保持反面向上，仍是一个二项分布，概率为$C_{N-i}^{X_u-m}p_{01}^{X_u-m}(1-p_{01})^{N-i-X_u+m}$。因此把两种情况相乘并对m求和即得到$P(X_u\,|\,A=i)$，m的求和范围可由组合数的定义得到。
分子代入$i=0$即可验证。
）

根据RAPPOR机制，

$$B_i'=\begin{cases}1, & \text{with probability } \frac{1}{2}f\\ 0, & \text{with probability } \frac{1}{2}f\\ B_i, & \text{with probability } 1-f\end{cases}\qquad P(S_i=1)=\begin{cases}q, & \text{if } B_i'=1.\\ p, & \text{if } B_i'=0.\end{cases}$$

可以计算出p_{01}(B中对应位为0翻转为S中对应位为1的概率)和p_{11}(B中对应位为1翻转为S中对应位为1的概率)：

$$p_{01}=p-0.5*f*(p-q),p_{00}=1-p_{01}$$
$$p_{11}=q+0.5*f*(p-q),p_{10}=1-p_{11}$$

先验概率ξ取为0.5。

优化函数为：

$$min \sum_{u=0}^{plc\_num} P(A=0|X_u)$$

三、优化算法

1）随机选取替换

首先选取candidate_num个样本，之后不断循环，每次循环将选中的人群中的一人和未选中的一人交换，计算后验概率和是否减小，如果减小，则完成替换，否则不替换（贪心算法）。如果经过一定次数的迭代，优化函数不变，则提前终止循环。

2）类似梯度下降的方法

将后验概率随X_u的变化关系连续化，通过线性回归得到指数关系的系数，之后可以求每个X_u对应的梯度。将已选中的志愿者的比特串和梯度向量点乘，得到移除后代价函数上升量的估计值，选取最小的一个。将未已选中的志愿者的比特串和梯度向量点乘，得到替换进来之后代价函数下降量的估计值，选取最大的一个。如果上升量小于下降量，则发生替换，否则不替换。实际仿真中，发现在代价函数趋于平缓时，导数会发生震荡，因此还需要设置最大迭代次数。

当plc_num较大时，后验概率出现下溢错误，只能通过较小X_u的函数关系做合理外推得到较大X_u处的导数。

四、数据集及仿真结果

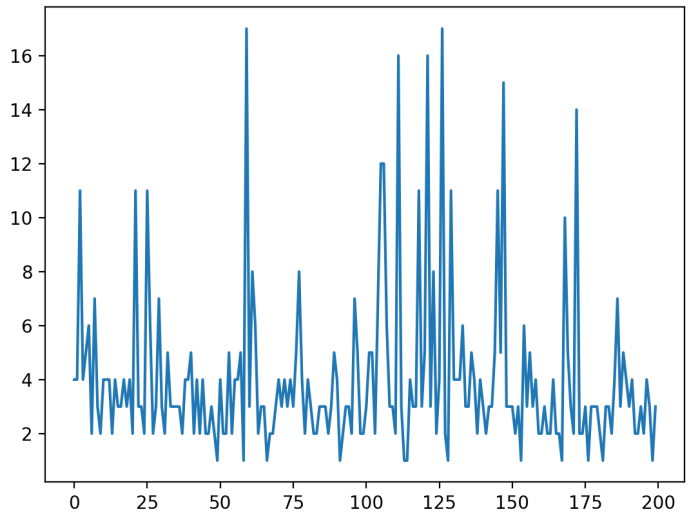
1）数据集构造

参数设置：

plc_num=200，people=500，candidate_num=50，k_favor=4，f=0.0

模拟真实场景中，部分地区访问率高，部分地区访问率低的情形。为保证对真实覆盖率上限的估计，人群由三类组成。第一类为全覆盖的2组人群（每组50人），如果选取其中任意一组，他们可以覆盖所有点。第二类为集中访问的人群（共150人），在20个地点中均匀分布。第三类（250人）在全空间均匀分布。

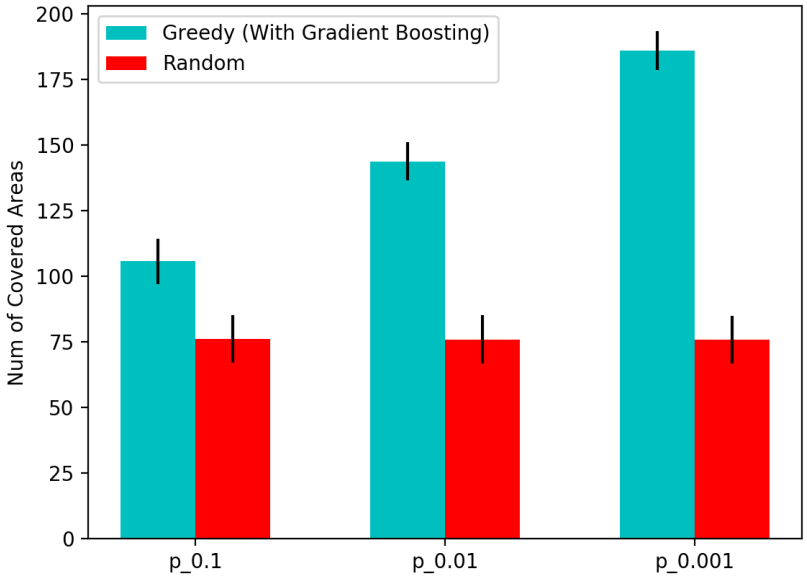
分布概况如下图所示：



3）仿真结果

调整转移概率，每个翻转概率对应20个随机生成的数据集，每个数据集上跑20次随机初始化的优化，与20次随机取样对比。

对于每个转移概率的仿真结果，计算均值和方差。



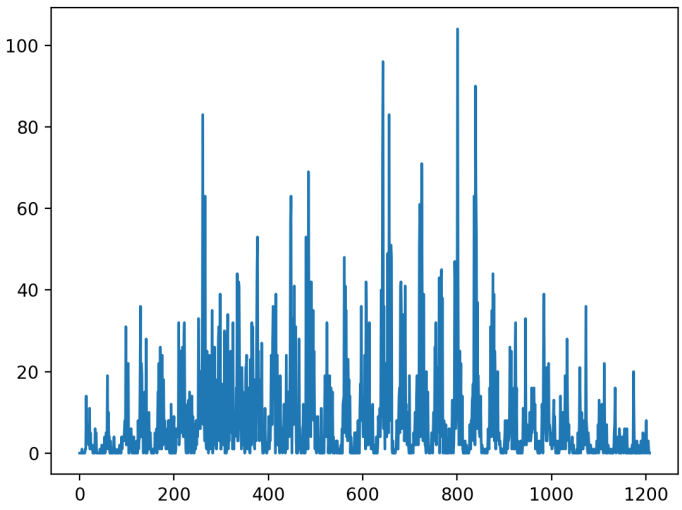
2）真实数据集

将经度、维度的粒度确定为0.01°，在经纬度中位数左右选取一定范围内的坐标作为用户访问记录。

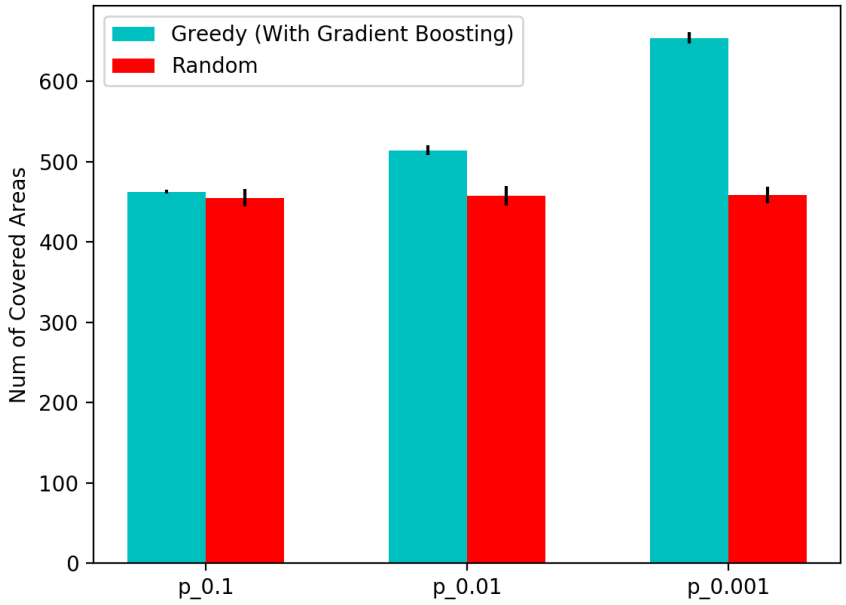
plc_num = 1209, people = 1978

candidate_num = 200, k_favor = 5

数据分布：



仿真结果：


Greedy (With Gradient Boosting)
Random
Num of Covered Areas
0
100
200
300
400
500
600
p_0.1
p_0.01
p_0.001

```
xrange=30, yrange=30, self.x_granu = 0.01,  self.y_granu = 0.01
eps = 4
p = 0.01
q = 1.0/((1.0-p)/(p*np.exp(eps)) + 1)
```

people = 4000 candidate = 1000 k_favor = 5	[(1597, 2129.2742942638024), (1589, 1589), (1511, 3600), (1537, 2173.5227883693569)] [(1600, 2129.0535559915133), (1568, 1568), (1485, 3600), (1528, 2173.2717401997897)]
people = 4000 candidate = 800 k_favor = 5	[(1479, 2235.3779651356313), (1506, 1506), (1403, 3599), (1384, 2284.9137197292321)] [(1466, 2234.2984796614751), (1475, 1475), (1413, 3600), (1362, 2285.7427549930408)]
people = 4000 candidate = 600 k_favor = 5	[(1326, 2373.8920405064864), (1361, 1361), (1224, 3598), (1226, 2430.9305070961177)] [(1310, 2373.9428839629136), (1378, 1378), (1217, 3599), (1186, 2431.2330623413595)]
people = 4000 candidate = 400 k_favor = 5	[(1049, 2570.4402198325406), (1146, 1146), (1014, 3585), (980, 2635.3454676299511)] [(1065, 2572.365849334527), (1145, 1145), (993, 3588), (1034, 2633.5655276068637)]
people = 4000 candidate = 200 k_favor = 5	[(639, 2888.9111280879788), (784, 784), (659, 3308), (667, 2957.3836307246975)] [(636, 2888.378526114795), (776, 776), (665, 3304), (660, 2957.426633336901)]

```
xrange=30, yrange=30, self.x_granu = 0.1,  self.y_granu = 0.1
eps = 4
p = 0.01
q = 1.0/((1.0-p)/(p*np.exp(eps)) + 1)
```

people = 3000 candidate = 1000 k_favor = 5	[(549, 3141.8996506567591), (809, 2791.0), (452, 3600), (450, 3169.9785663826301)]
people = 3000 candidate = 800 k_favor = 5	[(498, 3181.3482425172233), (809, 2791.0000007199806), (441, 3600), (418, 3212.4625409347846)]
people = 3000 candidate = 600 k_favor = 5	[(450, 3231.6899626970444), (406, 406), (360, 3598), (356, 3262.4246461962525)] [(436, 3230.213816584193), (430, 430), (358, 3599), (362, 3261.0145167085743)] [(434, 3232.1010567904427), (403, 403), (375, 3598), (354, 3263.9663804211755)]
people = 3000 candidate = 400 k_favor = 5	[(369, 3293.4979589287923), (369, 369), (275, 3585), (298, 3326.80781107241)]
people = 3000 candidate = 200 k_favor = 5	[(266, 3380.2339521175218), (284, 284), (200, 3280), (182, 3408.5308663975043)]

utility function graph
平滑先验 或者都趋于1

max result
优化空间大小

noisy prior
经验结果

1、优化空间与算法显著性比较

比较准则：计算各方法的覆盖率较之于random baseline的提升百分比

people = 3000, k_favor = 5, eps = 4

n%	no noisy*	our method	noisy baseline	random baseline
candidate=1000	80	22	0.4	0
candidate=800	94	19	6	0
candidate=600	134	21	-2	0
candidate=400	200	34	2	0
candidate=200	267	46	14	0

people = 3000, k_favor = 3, eps = 4

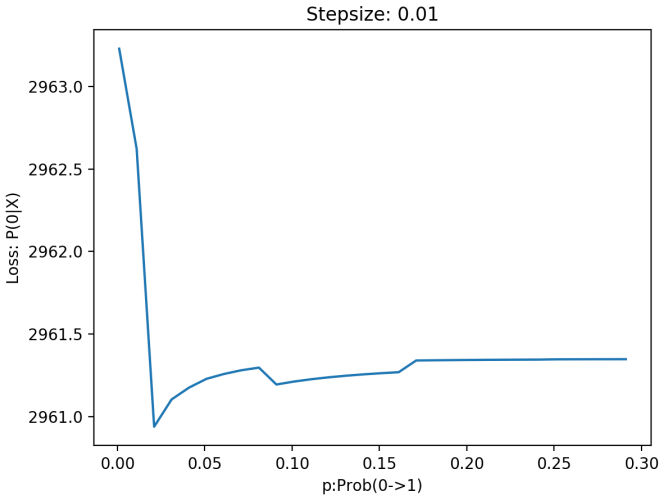
n%	no noisy*	our method	noisy baseline	random baseline
candidate=1000	74	17	-8	0
candidate=800	89	12	-10	0
candidate=600	130	23	-1	0
candidate=400	196	24	-7	0
candidate=200	282	47	-1	0

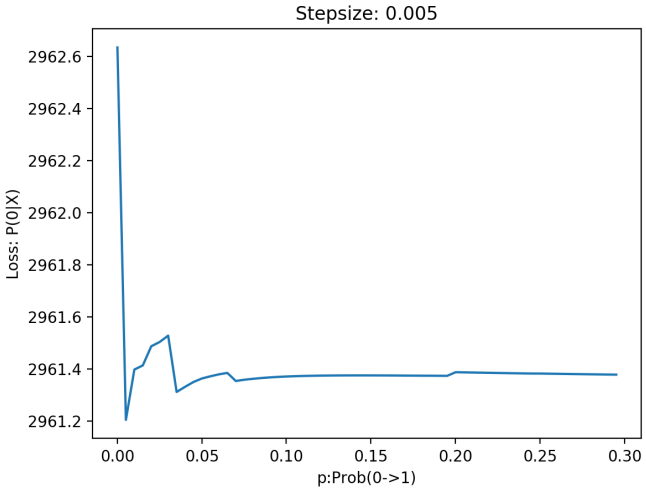
*：no noisy 方法用的是p=1e-5（p=0梯度无法计算，取较小数保证计算可进行），q=1，之后仍采用梯度下降的优化方法；较之于利用真实数据随机替换的策略，可以获得更高的覆盖率，可能原因是利用真实数据随机替换的策略需要更多的迭代次数才能收敛到一个比较好的解。

总体而言，随着优化空间的增大，目前算法较之于baseline的优势也在提升，但是和最优结果之间的差距也在逐渐增大。

2、utility function在大数据集上的表现

在服务器上开20个进程并行计算，需要1小时左右完成网格搜索。utility function选择为各地点后验概率向量和先验是否有人的0、1向量的RMSE；





选取两图中的最优点p=0.005,p=0.02以及两个极端值p=0.001,p=0.3进行训练。

n%	no noisy	our method	noisy baseline	random baseline
p=0.005	809	**529**	497	474
p=0.02	809	**528**	474	480
p=0.0001	809	460	464	458
p=0.2	809	510	450	471

可见，两种步长下所选取的最优值点在最终的覆盖率上都要优于其他在utility function上表现次优的点，说明utility function是可取的。