相向而行路人最优避让方案分析

KW

2022 年 12 月 15 日

摘要

分两种情况讨论：1. 路人两侧都有空档可以避让（向左避让、向右避让、直走）；2. 只有单侧有空档可以避让（只能向左/右避让、直走）。

1 路人两侧都有空档

	θ_1 ←	θ_2 ○	θ_3 →
θ_1 ↑	✓	✓	✗
θ_2 ○	✓	✗	✓
θ_3 ↓	✗	✓	✓

(1)

令向左避让概率、直走（或不动）概率、向右避让概率分别为 $\theta_1, \theta_2, \theta_3$。假设对二人来说各避让概率相同。此时各避让情况下发生碰撞的结果如图 1所示（勾表示无碰撞，叉表示碰撞）。令事件 Q 表示没有发生碰撞，则似然

$$p(Q \mid \boldsymbol{\theta}) = 1 - 2\theta_1\theta_3 - \theta_2^2$$

易知，$\theta_1 + \theta_2 + \theta_3 = 1$。于是可令先验概率

$$p(\boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \propto \prod_{i=1}^{3} \theta_i^{\alpha_i - 1}$$

其中 $\mathrm{Dir}(\cdot \mid \cdot)$ 为 Dirichlet 分布。因此，后验概率

$$p(\boldsymbol{\theta} \mid Q) \propto p(Q \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$$

两边取负对数

$$-\log p(\boldsymbol{\theta} \mid Q) \propto -\log(1-2\theta_1\theta_3-\theta_2^2) - \sum_{i=1}^{3}(\alpha_i - 1)\log\theta_i$$

为最大化后验概率，可得凸优化问题

$$\begin{aligned} \boldsymbol{\theta}^* = \ & \underset{\boldsymbol{\theta}}{\arg\min} \quad -\log(1-2\theta_1\theta_3-\theta_2^2) - \sum_{i=1}^{3}(\alpha_i - 1)\log\theta_i \\ & \text{s.t.} \qquad \sum_{i=1}^{3}\theta_i = 1, \\ & \qquad\quad 0 < \theta_i < 1 \quad \forall 1 \le i \le 3 \end{aligned} \tag{2}$$

1.1 求先验超参数

先验超参数 $\boldsymbol{\alpha}$ 可由调查问卷得到，即人们不刻意追求避免碰撞最优方案时的避让概率。经过包含 23 份问卷的调查，6 人选择向左避让，13 人选择向右避让，4 人选择直走或原地不动。使用 Laplace smoothing，7/26 概率向左避让，14/26 概率向右避让，5/26 概率直走或原地不动。令

$$\Theta_0 = \begin{pmatrix} 7/26 \\ 5/26 \\ 14/26 \end{pmatrix}$$

则

$$\boldsymbol{\alpha} = \Theta_0 / \min\Theta_0 = \begin{pmatrix} 7/5 \\ 1 \\ 14/5 \end{pmatrix}$$

1.2 求解最优方案

可用 SLSQP 算法求解。

$$\boldsymbol{\theta}^* = \begin{pmatrix} 1.01 \times 10^{-1} \\ 1.00 \times 10^{-12} \\ 8.99 \times 10^{-1} \end{pmatrix}$$

此时（避免碰撞的）后验概率为 0.82；注：先验概率为：0.67。也即，更优先向右避让能很大限度地避免碰撞。核心 Python 代码见 Listing 1。

Listing 1: 两侧有空档优化核心代码

```
import numpy as np
from scipy import optimize


def likelihood(x):
    return 1-2*x[0]*x[2]-x[1]**2

def obj(x, a):
    return -np.log(likelihood(x))-np.sum(a*np.log(x))


def jac(x, a):
    return 2*x[::-1]/likelihood(x)-a/x


def bounds(eps):
    return optimize.Bounds(0 + eps, 1 - eps, keep_feasible=True)


def constr():
    return [
        {
            'type': 'eq',
            'fun': lambda x: np.sum(x) - 1,
            'jac': lambda x: np.ones(3),
        },
    ]


# Success
```

```
def solve(theta0):
    theta0 = np.asarray(theta0)
    alpha = theta0 / np.min(theta0)
    a = alpha - 1
    x0 = np.random.rand(3)
    x0 /= np.sum(x0)
    res = optimize.minimize(obj, x0=x0, args=(a,),
                            method='SLSQP', jac=jac,
                            bounds=bounds(1e-12),
                            constraints=constr(),
                            options={'maxiter': 1000000})
    print('alpha:', alpha)
    print('Success:', res.success)
    if res.success:
        print('Solution:', res.x)
        print('Proba:', likelihood(res.x))
        print('Prior:', theta0)
        print('Prior proba:', likelihood(theta0))
```

2 单侧有空档

		θ_1 ○	θ_2 →
θ_1	○	✗	✓
θ_2	↑	✓	✗

(3)

令直走（或不动）概率、向空档方向避让概率分别为 θ_1 和 θ_2。假设对二人来说各避让概率相同。此时各避让情况下发生碰撞的结果如图 3所示（勾表示无碰撞，叉表示碰撞）。则似然

$$p(Q \mid \boldsymbol{\theta}) = 1 - \theta_1^2 - \theta_2^2$$

同理可得，后验概率的负对数

$$-\log p(\boldsymbol{\theta} \mid Q) \propto -log(1-\theta_1^2-\theta_2^2) - \sum_{i=1}^{2}(\alpha_i - 1)\log\theta_i$$

为最大化后验概率，可得凸优化问题

$$\begin{aligned} \boldsymbol{\theta}^* = \ & \arg\min_{\boldsymbol{\theta}} \quad -log(1-\theta_1^2-\theta_2^2) - \sum_{i=1}^{2}(\alpha_i - 1)\log\theta_i \\ & \text{s.t.} \qquad \sum_{i=1}^{2}\theta_i = 1, \\ & \qquad\quad 0 < \theta_i < 1 \quad \forall 1 \le i \le 2 \end{aligned} \tag{4}$$

2.1 求先验超参数

由于我没有特殊地做单侧避让的问卷，先验数据由1.1 节得到，并做如此假设：向空档方向避让的先验概率等于双侧空档情况下向左和向右避让先验概率之和。因此 5/26 概率直走或原地不动，21/26 概率向空档方向避让。同理可得

$$\boldsymbol{\alpha} = \begin{pmatrix} 1 \\ 21/5 \end{pmatrix}$$

2.2 求解最优方案

很可惜，没有最优方案。事实上，这符合直觉，因为图 3中勾和叉的分布是对称的。在当前先验数据下，不碰撞的先验概率为 0.31。无论采用何种避让方案，不碰撞的后验概率不会超过 0.31。附最优后验概率（即先验概率）

$$\boldsymbol{\theta}^* = \begin{pmatrix} 0.19 \\ 0.81 \end{pmatrix}$$

核心 Python 代码见 Listing 2。

Listing 2: 单侧有空档优化核心代码

```
import numpy as np
from scipy import optimize
```

```
def likelihood(x):
    return 1-np.sum(x**2)


def obj(x, a):
    return -np.log(likelihood(x))-np.sum(a*np.log(x))


def jac(x, a):
    return 2*x/likelihood(x)-a/x


def bounds(eps):
    return optimize.Bounds(0 + eps, 1 - eps, keep_feasible=True)


def constr():
    return [
        {
            'type': 'eq',
            'fun': lambda x: np.sum(x) - 1,
            'jac': lambda x: np.ones(2),
        },
    ]


# Success
def solve(theta0):
    theta0 = np.asarray(theta0)
    alpha = theta0 / np.min(theta0)
    a = alpha - 1
    x0 = np.random.rand(2)
    x0 /= np.sum(x0)
```

```
res = optimize.minimize(obj, x0=x0, args=(a,),
                        method='SLSQP', jac=jac,
                        bounds=bounds(1e-12),
                        constraints=constr(),
                        options={'maxiter': 1000000})
print('alpha:', alpha)
print('Success:', res.success)
if res.success:
    print('Solution:', res.x)
    print('Proba:', likelihood(res.x))
    print('Prior:', theta0)
    print('Prior proba:', likelihood(theta0))
```