統計的機械学習
第八回　レポート ID: 02

37-196360　森田涼介

2019 年 6 月 10 日

ある検査方法を分析するために被験者を集めることにした。陽性であった被験者を 5 人集めるために全部で 20 人の被験者を必要とした。少なくともあと 2 人陽性の被験者のデータを取るために，何人の被験者を集めれば良いかを考える。

事前分布を，a, b をパラメータとするベータ分布，尤度を負の二項分布とする。陽性となる確率を π とする Bernoulli 分布に従う n 回の独立した試行において，陽性となる回数が k となるまでに陰性となった回数を m とすると，

$$p(\pi) = \mathrm{Beta}(\pi|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\pi^{a-1}(1-\pi)^{b-1} \tag{1}$$

$$p(\mathrm{data}|\pi) = \mathrm{NB}(m|\pi) = \frac{n!}{k!(m+1)!}\pi^{k}(1-\pi)^{m} \tag{2}$$

$p(\mathrm{data})$ は定数であることに注意すると，π の事後分布の確率密度関数は，

$$p(\pi|\mathrm{data}) = \frac{p(\mathrm{data}|\pi)p(\pi)}{p(\mathrm{data})} \tag{3}$$

$$\propto p(\mathrm{data}|\pi)p(\pi) \tag{4}$$

$$= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\pi^{a-1}(1-\pi)^{b-1}\cdot\frac{n!}{k!(m+1)!}\pi^{k}(1-\pi)^{m} \tag{5}$$

$$\propto \pi^{a+k-1}(1-\pi)^{b+m-1} \tag{6}$$

これを正規化すると，結局，事後分布は，

$$p(\pi|\mathrm{data}) = \mathrm{Beta}(\pi|a+k,\ b+m) \tag{7}$$

となる。また，l 回の成功が得られるまでの失敗の数 x の期待値は，

$$E_{\mathrm{NB}(x|\pi)}[x] = l\frac{1-\pi}{\pi} \tag{8}$$

$$E[x|\mathrm{data}] = \int_0^1 E_{\mathrm{NB}(x|\pi)}[x]p(\pi|\mathrm{data})\mathrm{d}\pi = \int_0^1 l\frac{\Gamma(a+b+k+m)}{\Gamma(a+k)\Gamma(b+m)}\pi^{a+k-2}(1-\pi)^{b+m}\mathrm{d}\pi \tag{9}$$

となる。

$a=b=1$, $m=15$, $k=5$ のときを考える。あと $l=2$ 人陽性の人を集めるために平均的に必要な人数は，$l+E[x|\mathrm{data}]$ である。式 (9) を用いて，数値計算によりこれを求めると，

$$l+E[x|\mathrm{data}] = 2+6.4 = 8.4 \tag{10}$$

となる。

また，多く見積もることを考えると，π が比較的小さい値を取るとすればよい。例えば，$\pi \leq 0.1$ となるときを考えると，その確率は，

$$p(\pi \leq 0|\text{data}) = \int_0^{0.1} p(\pi|\text{data})\text{d}\pi = 0.0145 \tag{11}$$

$\pi = 0.1$ のとき，あと $l = 2$ 人陽性の人を集めるために必要な人数は，

$$l + E_{\text{NB}(x|\pi)}[x] = \frac{l}{\pi} = 20 \tag{12}$$

となる。

プログラム

Listings 1: assignment2.py

```python
# -*- coding: utf-8 -*-

import math
import numpy as np
import matplotlib.pyplot as plt


def beta_distribution(pi, a, b):
    value = (
        (math.gamma(a + b) / (math.gamma(a) * math.gamma(b)))
        * pi**(a-1) * (1 - pi)**(b-1)
        )
    return value


def make_beta(a, b):
    def _beta(pi):
        return beta_distribution(pi, a, b)
    return _beta


def expected_number(posterior_dist, l):
    def _num(pi):
        return l * ((1 - pi)/pi) * posterior_dist(pi)
    return _num


def integrate(distribution, lower=0.0, upper=1.0, dx=0.01,):
    x = np.arange(lower, upper+dx, dx)
    dist = distribution(x)
    p = (dist*dx).sum()
```

```
32      return p
33
34
35  def main():
36      # settings
37      k, m = 5, 15  # data
38      l = 2
39      a, b = 1, 1   # prior dist
40      threshold = 0.1
41      dx = 0.0001
42
43      # calc
44      posterior_dist = make_beta(a+k, b+m)
45      n_neg_expected = integrate(expected_number(posterior_dist, l), dx=dx,
        lower=dx)
46      print(f'a = {a}  b = {b}  E[neg] = {n_neg_expected:.2f}  #Expected =
        {l+n_neg_expected:.2f}')
47
48      p = integrate(posterior_dist, upper=threshold, dx=dx)
49      print(f'a = {a}  b = {b}  p(pi < {threshold}) = {p:.4f}')
50
51
52  if __name__ == '__main__':
53      main()
```