Stochastic Simulation Coursework 1

tmp120

November 2022

1 Derivations, Text and Figures

1.1 Question 1: sampling from chi-squared using rejection sampling

1.

Let $g(x) = p_v(x)/q_\lambda(x)$

Then $g(x) = \frac{x^{\nu/2-1}\exp^{(\lambda-1/2)x}}{\lambda 2^{\nu/2}\Gamma(\nu/2)}$

$\log(g(x)) = \log(\frac{1}{\lambda 2^{\nu/2}\Gamma(\nu/2)}) + (\lambda - 1/2)x + (\nu/2 - 1)\log(x)$

Let $\frac{\partial \log(g(x))}{\partial x} = (\lambda - 1/2) + \frac{\nu/2-1}{x} = 0.$

Solving for x, the optimal $x_\star = \frac{2-\nu}{2\lambda-1}$.

$\frac{\partial^2 \log(g(x))}{\partial^2 x} = \frac{1-\nu/2}{x^2} < 0$ as $x^2 > 0$ and $\nu > 2$ so $x_\star$ is a maximum.

Substituting in $x_\star$, $M_\lambda = \frac{(\frac{2-\nu}{2\lambda-1})^{\nu/2-1}\exp^{(1-\nu/2)}}{\lambda 2^{\nu/2}\Gamma(\nu/2)}$

2.

$M_\lambda = \frac{(\frac{2-\nu}{2\lambda-1})^{\nu/2-1}\exp^{(1-\nu/2)}}{\lambda 2^{\nu/2}\Gamma(\nu/2)}$

$\log(M_\lambda) = \log(\frac{1}{\lambda 2^{\nu/2}\Gamma(\nu/2)}) + (\nu/2-1)\log(2-\nu) - (\nu/2-1)\log(2\lambda-1) + (1-\nu/2)$

Let $\frac{\partial(\log(M_\lambda)}{\partial \lambda} = \frac{2(1-\nu/2)}{2\lambda-1} - \frac{1}{\lambda} = 0$

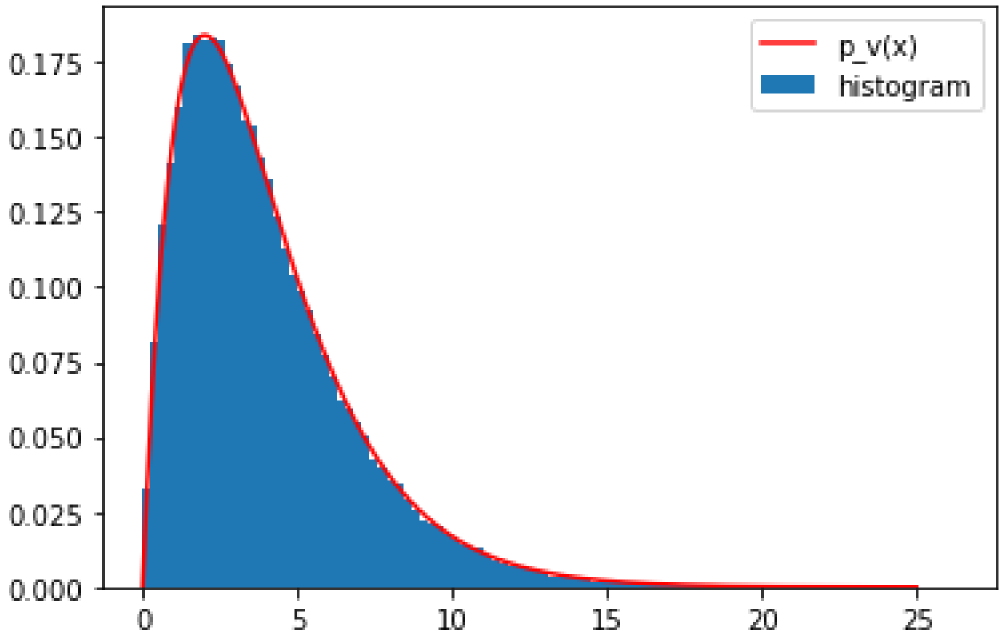
This solves for $\lambda_\star = \frac{1}{\nu}$

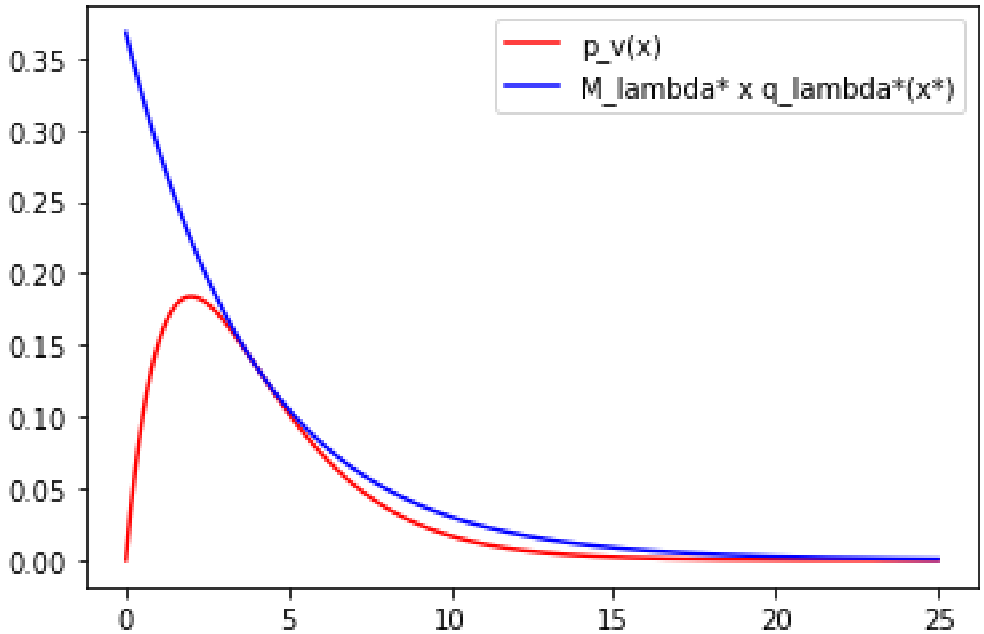
$\frac{\partial^2(\log(M_\lambda)}{\partial^2 \lambda} = \frac{2(\nu-2)}{(2\lambda-1)^2} + \frac{1}{\lambda^2} > 0$ again as $\nu > 2$ so $\frac{1}{\nu}$ gives the minimal value as required.

Hence $M_{\lambda_\star} = \frac{\nu^{\nu/2} \exp^{(1-\nu/2)}}{2^{\nu/2}\Gamma(\nu/2)}$.

3.

The rejection sampler leads to the following graphs.

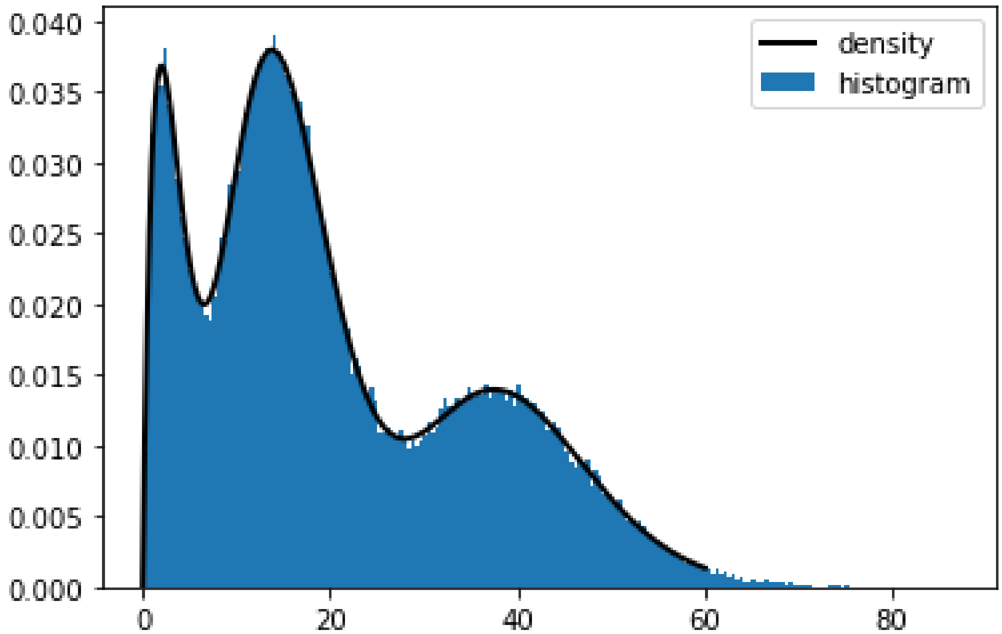




The empirical acceptance rate is 0.68097, which is very close to the theoretical acceptance rate 0.67957, $\hat{a}$, as expected.

1.2 Question 2: sample from a mixture of chi-squared

Sampling results in the following histogram and density.



2 Appendix

2.1 Code for question 1

```
import numpy as np
import matplotlib.pyplot as plt
def p(x, nu):
    return x ** (nu / 2 - 1) * np.exp(-x / 2) / (2 ** (nu / 2) *np.
                                        math. factorial (int(nu /2) -
                                         1))
def q(x, lamda):
    return lamda * np.exp(-lamda*x)
nu = 4
lamda = 1/nu
n = 100000
M = 1/(2**(nu/2)*np.math. factorial (int(nu /2) - 1))*nu**(nu/2)*np
                                  .exp(1-nu/2)
acc = 0 #number of accepted samples
for i in range(n):
    u1 = np.random.uniform(0, 1) # sample from uniform distribution
    x = -(1/lamda) * np.log(1 - u1) # inverse of the CDF
    u2 = np.random.uniform(0, 1) # uniform
    if u2 < p(x, nu)/(M * q(x, lamda)): # accept - reject
        x_samples = np.append(x_samples , x) # store sample if
                                         accepted
        acc += 1
xx = np.linspace(0, 25, 1000)
print(1/M) # theoretical acceptance rate
```

```
print(acc/n) # empirical acceptance rate
plt.hist(x_samples , bins = 100 , density=True)
plt.plot(xx , p(xx , nu), 'r-')
plt.legend(["p_v(x)","histogram"])
plt.show()
plt.plot(xx , p(xx , nu), 'r-')
plt.plot(xx , M*q(xx , lamda), 'b-')
plt.legend(["p_v(x)","M_lambda* x q_lambda*(x*)"])
plt.show ()
```

2.2 Code for question 2

```
def rejection_sample(nu):
    lamda = 1/nu
    while True:
        u1 = np.random.uniform(0, 1) # sample from uniform
                                       distribution
        x = -(1/lamda) * np.log(1 - u1) # inverse of the CDF
        u2 = np.random.uniform(0, 1) # uniform
        M = 1/(2**(nu/2)*np.math. factorial (int(nu /2) - 1))*nu**(
                                       nu/2)*np.exp(1-nu/2)
        if u2 < p(x, nu)/(M * q(x, lamda)): # accept - reject
            return x # store sample if accepted
n = 100000
v1 = 4; v2 = 16; v3 = 40
nu=np.array([v1,v2,v3])
w1 = 0.2; w2 = 0.5; w3 = 0.3
w = np.array([w1 , w2, w3])
def sample_discrete (w): # draws a single index (0,... ,K-1)
    cw = np.cumsum(w)
    sample = []
    u = np.random.uniform(0, 1)
    for k in range(len(cw)):
        if cw[k] > u:
            sample = k
            break
    return sample
x_samples = np.array([])
for i in range(n):
    samp = sample_discrete (w) # sample an index from the discrete
    if samp == 0: # if the index is 0, sample from first one
        x = rejection_sample(nu[0])
    elif samp== 1: # if the index is 1, sample from the second one
        x = rejection_sample(nu[1])
    elif  samp==2:
        x=rejection_sample(nu[2])
    x_samples = np.append(x_samples , x)
def mixture_density (x, w, nu):
    return w[0]*p(x, nu[0]) + w[1]*p(x, nu[1]) + w[2]*p(x, nu[2])
xx = np.linspace(0, 60 , 1000)
plt.plot(xx , mixture_density (xx , w, nu), color='k', linewidth=2)
plt.hist(x_samples , bins = 200 , density=True)
plt.legend(["density","histogram"])
plt.show()
```