Coursework 1

Imperial College London

Department of Mathematics

Stochastic Simulation

Author: 01844579

Date: November 7, 2022

1 Question 1

1.1 Question 1.1

To compute $M_\lambda = \sup_x p_\nu(x)/q_\lambda(x)$ we will first optimise over x and find x^* and then substitute it into our expression. First we let,

$$R(x) = \frac{p_\nu(x)}{q_\lambda(x)} = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)} \cdot \frac{1}{\lambda} \cdot x^{\frac{\nu}{2}-1} \cdot e^{-\frac{x}{2}} \cdot e^{\lambda x} = \frac{1}{2^{\frac{\nu}{2}}} \cdot \frac{1}{\Gamma\left(\frac{\nu}{2}\right)} \cdot \frac{1}{\lambda} \cdot x^{\frac{\nu}{2}-1} \cdot e^{x\left(\lambda - \frac{1}{2}\right)}.$$

To find the turning points of $R(x)$ it is easier to first take the logarithm,

$$\log(R(x)) = \log\left(\frac{1}{2^{\frac{\nu}{2}}} \cdot \frac{1}{\Gamma\left(\frac{\nu}{2}\right)} \cdot \frac{1}{\lambda}\right) + \left(\frac{\nu}{2} - 1\right)\log(x) + x\left(\lambda - \frac{x}{2}\right),$$

and then differentiate

$$\frac{d\log(R(x))}{dx} = \frac{1}{x}\left(\frac{\nu}{2} - 1\right) + \lambda - \frac{1}{2}.$$

Setting the derivative equal to zero,

$$\frac{1}{x}\left(\frac{\nu}{2} - 1\right) + \lambda - \frac{1}{2} = 0 \implies \frac{1}{x}(\nu - 2) = 1 - 2\lambda.$$

We find that $x^* = \frac{\nu-2}{1-2\lambda}$. To check x^* is the optimal value of x we find the second derivative of $\log(R(x))$.

$$\frac{d^2\log(R(x))}{dx^2} = -\frac{1}{x^2}\left(\frac{\nu}{2} - 1\right)$$

Upon substituting $x^* = \frac{\nu-2}{1-2\lambda}$ we get

$$\frac{d^2\log(R(x^*))}{dx^2} = -\left(\frac{1-2\lambda}{\nu-2}\right)^2\left(\frac{\nu}{2} - 1\right)$$

Noticing that our first term is strictly positive since $0 < \lambda < 1/2$ and that we have that $\nu/2 - 1 > 0$ since we assumed that $\nu > 2$. So our second derivative is strictly smaller than 0 and therefore a maximum. Now that we have found x^* and proved it is indeed the maximum, we substitute it into M_λ. So our expression reads

$$M_\lambda = \sup_x \frac{p_\nu(x)}{q_\lambda(x)} = \frac{p_\nu(x^*)}{q_\lambda(x^*)} = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)\lambda} \cdot \left(\frac{\nu-2}{1-2\lambda}\right)^{\frac{\nu}{2}-1} \cdot e^{\left(\lambda-\frac{1}{2}\right)\left(\frac{\nu-2}{1-2\lambda}\right)}.$$

1.2 Question 1.2

After finding x^* and an expression for M_λ we want to now find the optimal λ^*. To do so we first take the logarithm of M_λ

$$\begin{aligned}\log(M_\lambda) &= \log\left(\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)\lambda}\right) + \left(\frac{\nu}{2} - 1\right)\log\left(\frac{\nu-2}{1-2\lambda}\right) + \left(\lambda - \frac{1}{2}\right)\left(\frac{\nu-2}{1-2\lambda}\right) \\ &= \log\left(\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}\right) - \log(\lambda) + \left(\frac{\nu}{2} - 1\right)\log(\nu - 2) - \left(\frac{\nu}{2} - 1\right)\log(1 - 2\lambda) + (\nu - 2)\left(\frac{\lambda - \frac{1}{2}}{1-2\lambda}\right)\end{aligned}$$

Now differentiating our expression with respect to λ

$$\frac{d\log(M_\lambda)}{d\lambda} = -\frac{1}{\lambda} + \left(\frac{\nu}{2} - 1\right)\frac{2}{1-2\lambda} + (\nu-2)\left(\frac{(1-2\lambda)+2\left(\lambda-\frac{1}{2}\right)}{(1-2\lambda)^2}\right).$$

Noticing that the last term is equal to 0 and setting the expression equal to 0 we obtain

$$-\frac{1}{\lambda} + \frac{\nu-2}{1-2\lambda} = 0 \implies \lambda^* = \frac{1}{\nu}.$$

To check λ^* indeed minimises M_λ we find that the second derivative of $\log(M_\lambda)$ is

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

Plugging in our value of $\lambda^* = 1/\nu$ we obtain

$$\frac{d^2\log(M_{\lambda^*})}{d\lambda^2} = \nu^2 + \frac{2\nu^2(\nu-2)}{\nu^2-4\nu+4} = \frac{\nu^2\left(\nu^2-4\nu+4\right)+2\nu^2(\nu-2)}{\nu^2-4\nu+4} = \frac{\nu^3(\nu-2)}{(\nu-2)^2} = \frac{\nu^3}{\nu-2}.$$

Note that our final result is strictly greater than 0 since we have that $\nu > 2$, therefore $\lambda^* = 1/\nu$ minimises M_λ.

1.3 Question 1.3

Figure 1 shows a histogram of our rejection sampler for $\nu = 4$ with the corresponding optimal λ. The theoretical acceptance rate is $\hat{a} = 0.6796$ and the calculated acceptance rate for Figure 1 is $a = 0.6816$. As we can see the acceptance rates obtained are very close values to one another, demonstrated by the histogram and density plot.

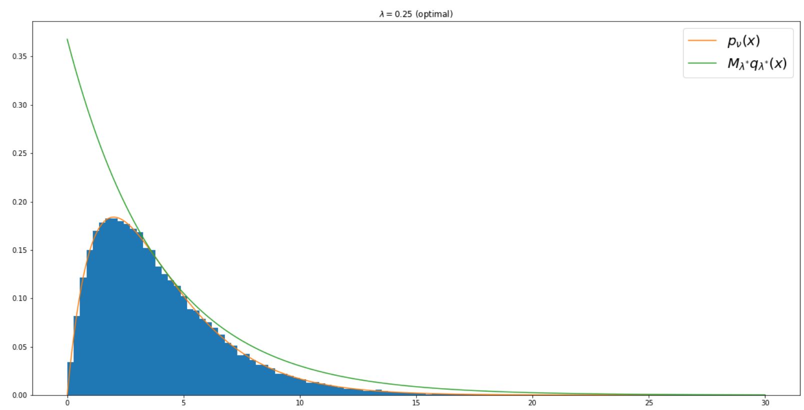

Figure 1: Rejection sampling procedure with optimal $\lambda = 1/\nu = 0.25$ for $n = 100000$.

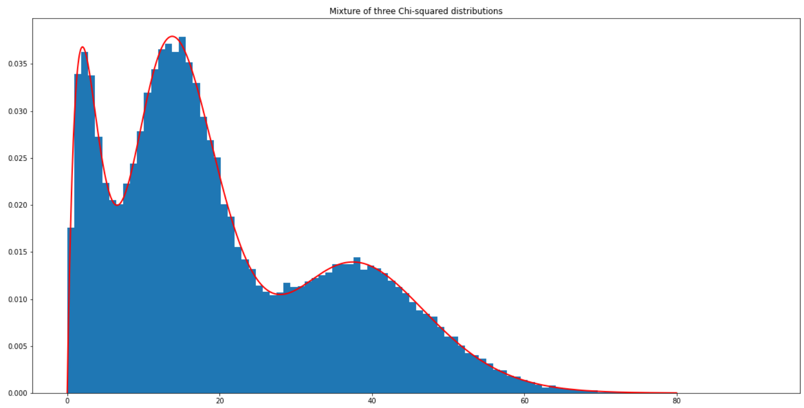

Figure 2: Histogram and density plot of mixture of Chi-squared distributions

2 Question 2

Figure 2 shows sampling from discrete mixture of Chi-squared distributions. First, indices are sampled from a discrete distribution using the inversion method. Then modifying our sampling previous rejection sampler we can take samples of respective Chi-squared distributions.

A Code for Question 1

```
n = 100000 # Number of samples to draw
nu = np.array([4, 16, 40]) # Array of nu values (will only use first
    element for Q1)
lam = 1/nu # Lambda values (will only use first element for Q1)
count = 0 # Counter to calculate accepted samples
w = [0.2, 0.5, 0.3] # Weights of discrete distribution
s = [0, 1, 2] # Support of discrete distribution

x_samples = [] # Empty list to append samples from exponential
    distribution
samples = [] # Empty list to append samples from a rejection sampling
     procedure

# Defining functions
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, lam):
    return lam*np.exp(-lam * x)

def m(nu):
    return nu * nu ** (nu / 2 - 1) * np.exp(1 - nu / 2) / (2 ** (nu /
     2) * np.math.factorial(int(nu / 2) - 1))
```

```

m = np.array([m(nu[0]), m(nu[1]), m(nu[2])]) # M values for acceptance probabilities (derived)

for i in range(n): # Code to sample from exponential distribution using inversion

    a = np.random.uniform(0,1)
    b = -(1/lam[0]) * np.log(1-a)
    x_samples = np.append(x_samples, b)

for i in range(n): # Rejection algorithm

    x = x_samples[i] # Proposal
    u = np.random.uniform(0,1) # Uniform
    acceptance = p(x, nu[0])/(m[0]*q(x, lam[0])) # Acceptance probability

    if u < acceptance: # Accept/reject
        samples = np.append(samples, x) # Store accepted values
        count += 1 # Increase count for accepted samples (for acceptance rate)

fig = plt.figure(figsize=(20,10)) # Code for plotting histogram and densities
x = np.linspace(0, 30, 1000)
plt.hist(samples,bins=100, density =True)
plt.plot(x, p(x, nu[0]), label = r'$p_{\nu}(x)$')
plt.plot(x, m[0]*q(x,lam[0]), label = r'$M_{\lambda^{*}} q_{\lambda^{*}}(x)$')
plt.legend(fontsize=20)
plt.title(r'$\lambda = 0.25$ (optimal)')
plt.show()

# Calculations for acceptance rates
print(f"Acceptance Rate: {count/n}")
print(f"Theoretical Acceptance Rate: {1/m[0]}")
fig.savefig('q1.png')
```

B Code for Question 2

```
def rejection_sampler(nu, lam, m): # Modified previous rejection sampler to obtain a single sample

    sample = []

    while len(sample) < 1:

        a = np.random.uniform(0,1)
        x = -(1/lam) * np.log(1-a)
        u = np.random.uniform(0,1)
        acceptance = p(x, nu)/(m*q(x, lam))

```

```
        if u < acceptance:
            sample = np.append(sample, x)
    return sample

def discrete_sampler(s, w): # Samples from discrete distribution defined on s with probabilities w (for indicies)

    cdf = np.cumsum(w)
    sample = []
    u = np.random.uniform(0,1)

    for k in range(len(cdf)):
        if cdf[k] > u:
            sample = s[k]
            break
    return sample

samples = [] # Empty list to append accepted samples

for i in range(n): # Samples from mixture of distributions

    u = discrete_sampler(s,w) # Sample from discrete distribution
    samples = np.append(samples, rejection_sampler(nu[u],lam[u], m[u])) # Samples from appropiate Chi-squared

def mixture_density(x, w, nu): # Code for plotting density
    return w[0]*p(x,nu[0]) + w[1]*p(x,nu[1]) + w[2]*p(x,nu[2])

figg = plt.figure(figsize=(20,10)) # Code for plotting histogram and density
xx = np.linspace(0, 80, 1000)
plt.hist(samples, bins =100, density=True)
plt.plot(xx, mixture_density(xx, w, nu), color='r', linewidth=2)
plt.title('Mixture of three Chi-squared distributions')
plt.show()
figg.savefig('q2.png')
```