Assignment 1

Reinforcement Learning KU, WS 2024/25

Team Members		
Last name	First name	Matriculation Number
Hinum-Wagner	Jakob	01430617

Task 1 - Monte Carlo Simulation [7.5 Points]

1

. A random variable X with state space $\mathcal{X} = \{0, 1, 2, 3, 4\}$ has the following probability mass function:

$$p(X = k) = \begin{cases} M \cdot \frac{\lambda^k}{k!} & \text{for } k = 0, 1, 2, 3, 4 \\ 0 & \text{otherwise} \end{cases}$$

where $k!$ is the factorial of k and M is a normalizing constant. Let $\lambda = 4$.

(a) Find the normalizing constant M such that $\sum_{k \in \mathcal{X}} p(X = k) = 1$.

(b) Find $E_X[X]$.

(c) Use Monte Carlo simulation to approximate the expected value of X and show a convergence plot with the y-axis representing the estimated expected value and the x-axis the number of simulations utilized to calculate the estimated expected value. Use 100, 200, 300, . . . , 10000 simulations and obtain an estimate for each value. Draw a horizontal line representing the exact expectation. Include this plot in your report.

0.1 Solution to Task 1 a): Find the normalizing constant M such that $\sum_{k \in \mathcal{X}} p(X = k) = 1$

To find the normalizing constant M so that the probability mass function sums to 1, I need to compute:

$$\sum_{k=0}^{4} p(X = k) = 1$$

Given $p(X = k) = M \cdot \frac{\lambda^k e^{-\lambda}}{k!}$ with $\lambda = 4$, this equation becomes:

$$M \cdot e^{-4} \sum_{k=0}^{4} \frac{4^k}{k!} = 1$$

Now, I'll calculate the sum:

$$S = \sum_{k=0}^{4} \frac{4^k}{k!} = \frac{4^0}{0!} + \frac{4^1}{1!} + \frac{4^2}{2!} + \frac{4^3}{3!} + \frac{4^4}{4!}$$

Breaking it down, I get:

$$S = 1 + 4 + \frac{16}{2} + \frac{64}{6} + \frac{256}{24}$$

After simplifying, I find:

$$S = 1 + 4 + 8 + \frac{32}{3} + \frac{32}{3} = 13 + \frac{64}{3} = \frac{103}{3}$$

Now, I set up the equation for M:

$$M \cdot e^{-4} \cdot \frac{103}{3} = 1$$

Solving for M, I get:

$$M = \frac{3 \cdot e^4}{103}$$

0.2 Solution to Task 1 b) Find $E_X[X]$.

To determine the expected value $E[X]$ of the random variable X, I start by using the definition of the expected value for discrete random variables:

$$E[X] = \sum_{k=0}^{4} k \cdot p(X = k)$$

Given the probability mass function (PMF):

$$p(X = k) = M \cdot \frac{\lambda^k e^{-\lambda}}{k!} \quad \text{for } k = 0, 1, 2, 3, 4$$

with $\lambda = 4$ and the normalizing constant M that I found earlier:

$$M = \frac{3e^4}{103}$$

First, I observe that $e^{-\lambda}$ is included in the PMF, so when calculating $M \cdot e^{-\lambda}$, it simplifies as follows:

$$M \cdot e^{-4} = \frac{3e^4}{103} \cdot e^{-4} = \frac{3}{103}$$

This allows me to rewrite the PMF as:

$$p(X = k) = \frac{3}{103} \cdot \frac{4^k}{k!}$$

Now, I proceed by computing each term $k \cdot p(X = k)$ for $k = 0$ to 4:

- For $k = 0$:
$$0 \cdot p(0) = 0 \cdot \left(\frac{3}{103} \cdot \frac{4^0}{0!} \right) = 0$$
- For $k = 1$:
$$1 \cdot p(1) = 1 \cdot \left(\frac{3}{103} \cdot \frac{4^1}{1!} \right) = \frac{12}{103}$$
- For $k = 2$:
$$2 \cdot p(2) = 2 \cdot \left(\frac{3}{103} \cdot \frac{4^2}{2!} \right) = 2 \cdot \frac{48}{103} = \frac{96}{103}$$
- For $k = 3$:
$$3 \cdot p(3) = 3 \cdot \left(\frac{3}{103} \cdot \frac{4^3}{3!} \right) = 3 \cdot \frac{64}{103} = \frac{192}{103}$$
- For $k = 4$:
$$4 \cdot p(4) = 4 \cdot \left(\frac{3}{103} \cdot \frac{4^4}{4!} \right) = 4 \cdot \frac{128}{103} = \frac{384}{103}$$

I then sum these values to find $E[X]$:
$$E[X] = 0 + \frac{12}{103} + \frac{96}{103} + \frac{192}{103} + \frac{384}{103} = \frac{684}{103}$$
To approximate, I get:
$$E[X] = \frac{684}{103} \approx 6.64$$
Thus, the expected value $E[X]$ is approximately 2.757.

0.3 Solution to Task 1 c) Use Monte Carlo simulation to approximate the expected value of X and show a convergence plot with the y-axis representing the estimated expected value and the x-axis the number of simulations utilized to calculate the estimated expected value. Use 100, 200, 300, . . . , 10000 simulations and obtain an estimate for each value. Draw a horizontal line representing the exact expectation. Include this plot in your report.

To approximate the expected value of the random variable X using Monte Carlo simulation, I followed a structured approach, visualizing the convergence of the estimate as the number of simulations increased. Here's a breakdown of the steps I took:

1. **Compute the Probabilities:** I precomputed the probabilities $p(X = k)$ for $k = 0, 1, 2, 3, 4$ using the provided probability mass function (PMF). This ensures that the probabilities are correctly normalized and sum to 1.

2. **Generate Samples:** To generate samples from the discrete distribution, I used the inverse transform sampling method, which is effective for discrete distributions. For each sample, I generated a random number r between 0 and 1, then determined the value of X by finding the smallest k such that $r \leq \text{CDF}(k)$, where CDF represents the cumulative distribution function.

3. **Estimate Expected Value:** For each specified number of simulations $N = 100, 200, \ldots, 10000$, I calculated the sample mean $\hat{E}[X]$, which serves as an estimate of the expected value. This approach allows me to observe how the estimate converges as N increases.

4. **Plot Convergence:** Finally, I plotted $\hat{E}[X]$ against N and included a horizontal line representing the exact expected value $E[X] = \frac{284}{103}$. This plot helps to visualize the convergence behavior of the estimate.

From the plot, I can observe the behavior of the estimated expected value $\hat{E}[X]$ as the number of simulations increases. Initially, with a smaller number of simulations, the estimated expected value fluctuates significantly around the exact expected value. This variability is expected due to the limited sample size, which introduces a higher degree of randomness in each estimate.

As the number of simulations grows, the fluctuations gradually decrease, and the estimated value converges closer to the exact expected value of $E[X] = 2.757$. This trend illustrates the law of large numbers, where increasing the sample size leads to a more accurate approximation of the true expected value. In this case, by around 10,000 simulations, the estimated expected value is consistently near the exact value, with only minor deviations.

The horizontal dashed line at $E[X] = 2.757$ serves as a reference, highlighting how the estimate stabilizes around this value. This convergence demonstrates that the Monte Carlo simulation method can provide a reliable approximation of the expected value, especially as the number of simulations becomes sufficiently large.

2

The probability density function of a continuous random variable X is given by:

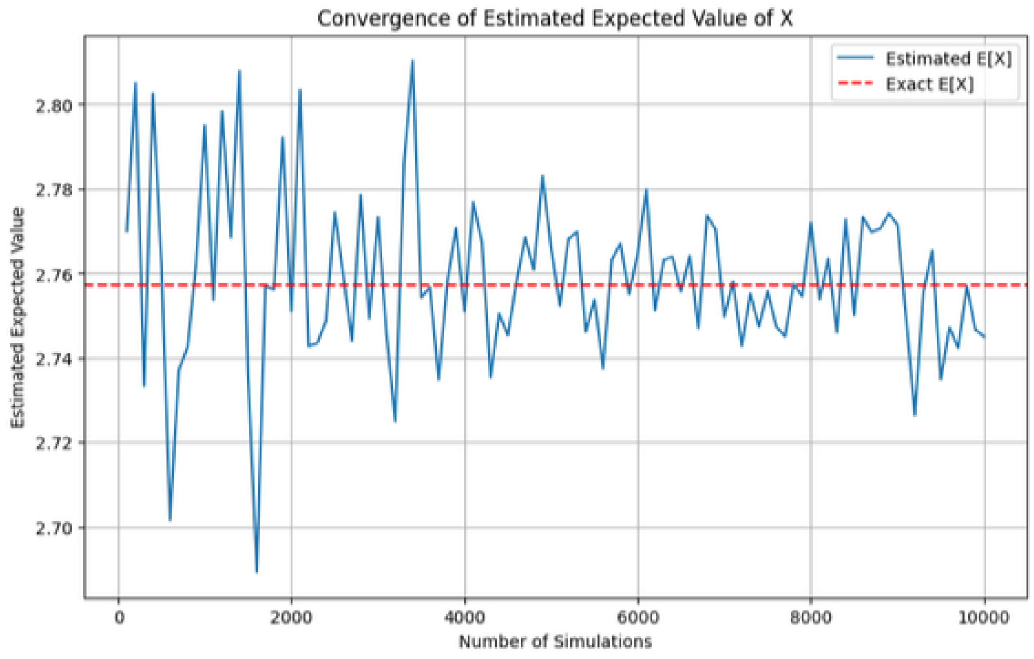


Figure 1: Convergence of Estimated Expected Value of X as the Number of Simulations Increases. The blue line represents the estimated expected value, while the red dashed line indicates the exact expected value $E[X] = \frac{284}{103} \approx 2.757$.

$$p(x) = \begin{cases} \frac{x}{4} & 0 < x < 2 \\ 0 & \text{otherwise} \end{cases}$$

(a) Analytically compute $E_X[X]$.

(b) The corresponding *cumulative distribution function* (CDF) is defined as

$$F(x) = \int_{-\infty}^{x} p(z)\, dz.$$

Write down $F(x)$ as a simple function of x that does not involve an integral (i.e., solve the definite integral).

(c) We can sample from $p(x)$ using the *inverse transform sampling* trick: First, sample $u \sim \text{Unif}(0, 1)$, and then, compute $x = F^{-1}(u)$ where F^{-1} denotes the inverse of F. The result x is a proper sample from p. Write down $F^{-1}(u)$ and implement this sampling procedure.

(d) Use this sampling procedure to estimate $E[X]$ via Monte Carlo for 100, 200, 300, ..., 10000 simulations. Similar to Task 1.1, show a convergence plot where you draw a horizontal line at the true expectation. Include this plot in your report.

0.4 Solution to Task 2 a): Analytically compute $E_X[X]$

To compute the expected value $E[X]$ of the continuous random variable X with the given probability density function (pdf):

$$p(x) = \begin{cases} \frac{x}{2} & \text{for } 0 < x < 2 \\ 0 & \text{otherwise} \end{cases}$$

I use the definition of the expected value for continuous random variables:

$$E[X] = \int_{-\infty}^{\infty} x \cdot p(x)\, dx$$

Since $p(x)$ is nonzero only on the interval $(0, 2)$, the integral simplifies to:

$$E[X] = \int_0^2 x \cdot p(x)\, dx = \int_0^2 x \cdot \frac{x}{2}\, dx$$

I simplify the integrand as follows:

$$x \cdot \frac{x}{2} = \frac{x^2}{2}$$

So the expected value becomes:

$$E[X] = \frac{1}{2} \int_0^2 x^2\, dx$$

Now, I compute the integral step by step:

- First, I integrate x^2:

$$\int x^2\, dx = \frac{x^3}{3}$$

- Then, I evaluate the definite integral from 0 to 2:

$$\int_0^2 x^2\, dx = \left[\frac{x^3}{3}\right]_0^2 = \frac{2^3}{3} - \frac{0^3}{3} = \frac{8}{3}$$

- Finally, I multiply by the constant $\frac{1}{2}$:

$$E[X] = \frac{1}{2} \times \frac{8}{3} = \frac{4}{3}$$

Answer: $E[X] = \frac{4}{3}$

0.5 Solution to Task 2 b): Write down $F(x)$ as a simple function of x that does not involve an integral (i.e., solve the definite integral)

To find the cumulative distribution function (CDF) $F(x)$ corresponding to the given probability density function (pdf):

$$p(x) = \begin{cases} \frac{x}{2} & \text{for } 0 < x < 2 \\ 0 & \text{otherwise} \end{cases}$$

I use the definition of the CDF for a continuous random variable:

$$F(x) = \int_{-\infty}^{x} p(z)\, dz$$

My goal is to express $F(x)$ as a simple function of x without integrals. I'll examine different intervals of x to cover the entire real line.

1. For $x \leq 0$:

   Since $p(z) = 0$ for $z \leq 0$, the CDF for any $x \leq 0$ is:

$$F(x) = \int_{-\infty}^{x} p(z)\, dz = \int_{-\infty}^{x} 0\, dz = 0$$

   So,

$$F(x) = 0 \quad \text{for } x \leq 0$$

2. For $0 < x < 2$:

   In this interval, $p(z) = \frac{z}{2}$. Therefore, I compute:

$$F(x) = \int_{-\infty}^{x} p(z)\, dz = \int_{-\infty}^{0} p(z)\, dz + \int_{0}^{x} p(z)\, dz = 0 + \int_{0}^{x} \frac{z}{2}\, dz$$

   Now, I calculate the integral:

$$F(x) = \frac{1}{2}\int_0^x z\,dz = \frac{1}{2}\left[\frac{z^2}{2}\right]_0^x = \frac{1}{2}\cdot\frac{x^2}{2} = \frac{x^2}{4}$$

Thus,

$$F(x) = \frac{x^2}{4} \quad \text{for } 0 < x < 2$$

3. For $x \geq 2$:

   For $z \geq 2$, the pdf $p(z) = 0$, meaning the CDF reaches a constant value for $x \geq 2$. I find this constant by computing $F(2)$:

   $$F(2) = \int_{-\infty}^2 p(z)\,dz = \int_0^2 \frac{z}{2}\,dz = \frac{1}{2}\left[\frac{z^2}{2}\right]_0^2 = \frac{1}{2}\cdot\frac{4}{2} = \frac{4}{4} = 1$$

   Since the total area under the pdf is 1, for any $x \geq 2$, the CDF remains at 1:

   $$F(x) = 1 \quad \text{for } x \geq 2$$

Summary:
Combining all intervals, the cumulative distribution function $F(x)$ is:

$$F(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ \frac{x^2}{4} & \text{for } 0 < x < 2 \\ 1 & \text{for } x \geq 2 \end{cases}$$

Answer:

$$F(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ \frac{x^2}{4} & \text{for } 0 < x < 2 \\ 1 & \text{for } x \geq 2 \end{cases}$$

0.6 Solution to Task 2 c): We can sample from $p(x)$ using the *inverse transform sampling* trick: First, sample $u \sim$ Unif$(0, 1)$, and then, compute $x = F^{-1}(u)$ where F^{-1} denotes the inverse of F. The result x is a proper sample from p. Write down $F^{-1}(u)$ and implement this sampling procedure.

To sample from the probability density function $p(x)$ using the inverse transform sampling method, I need to find the inverse of the cumulative distribution function (CDF), $F^{-1}(u)$, where u is a uniform random variable on the interval $[0, 1]$.

Given:

The cumulative distribution function $F(x)$ is:

$$F(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ \frac{x^2}{4} & \text{for } 0 < x < 2 \\ 1 & \text{for } x \geq 2 \end{cases}$$

Finding $F^{-1}(u)$ and Implementing the Sampling Procedure

Step 1: Find $F^{-1}(u)$

I need to solve for x in terms of u when $0 < x < 2$ and $0 < u < 1$. Starting with:

$$u = F(x) = \frac{x^2}{4}$$

I solve for x:

$$x^2 = 4u$$

Since $x > 0$, I take the square root on both sides:

$$x = 2\sqrt{u}$$

Thus, the inverse CDF $F^{-1}(u)$ is:

$$F^{-1}(u) = 2\sqrt{u} \quad \text{for } 0 \leq u \leq 1$$

Step 2: Implement the Sampling Procedure

To generate samples from $p(x)$, I follow these steps:

1. Sample u from a uniform distribution on $[0, 1]$:

$$u \sim \text{Uniform}(0, 1)$$

2. Compute x using the inverse CDF:

$$x = F^{-1}(u) = 2\sqrt{u}$$

The resulting x is a sample from the distribution defined by $p(x)$.
Explanation of the Code

- **Function sample_from_p():** This function generates a random number u uniformly between 0 and 1. Using the inverse transform sampling, it then computes $x = 2\sqrt{u}$ to obtain a sample from the probability density function $p(x)$.
- **Generating Samples:** We generate a specified number of samples (e.g., 10,000) by repeatedly calling the `sample_from_p()` function. These samples represent data points drawn from the distribution defined by $p(x)$.
- **Visualization:** A histogram of the sampled data is plotted to visualize the empirical distribution of x. To verify that the sampling was done correctly, the theoretical probability density function $p(x) = \frac{x}{2}$ is overlaid as a red line on the histogram for comparison.

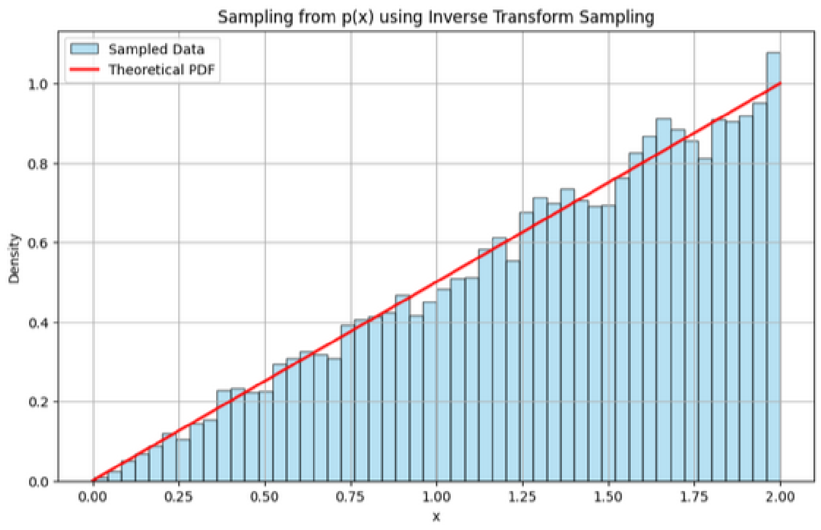


Figure 2: Empirical Distribution of Sampled Data from $p(x)$ Compared to Theoretical PDF. The blue histogram represents the distribution of the sampled data, while the red line shows the theoretical PDF $p(x) = \frac{x}{2}$.

In the plot, the histogram represents the distribution of the sampled data generated using inverse transform sampling. The shape of the histogram closely aligns with the theoretical probability density function $p(x) = \frac{x}{2}$, which is overlaid as a red line. This agreement between the histogram and the theoretical curve confirms that the sampling procedure accurately reproduces the desired distribution.

The plot shows that as x increases, the density of the samples increases linearly, consistent with the functional form of $p(x) = \frac{x}{2}$. The alignment of the histogram bars with the red line indicates that the inverse transform sampling method is effective for generating samples from $p(x)$. This method allows us to recreate the underlying distribution from which we are sampling.

0.7 Solution to Task 2 d): Use this sampling procedure to estimate $E[X]$ via Monte Carlo for 100, 200, 300, ..., 10000 simulations. Similar to Task 1.1, show a convergence plot where you draw a horizontal line at the true expectation. Include this plot in your report.

Explanation of the Code

- **Function sample_from_p():** This function generates a random number u uniformly between 0 and 1. Using the inverse transform sampling, it then computes $x = 2\sqrt{u}$ to obtain a sample from the probability density function $p(x)$.
- **Generating Samples:** We generate a specified number of samples (e.g., 10,000) by repeatedly calling the `sample_from_p()` function. These samples represent data points drawn from the distribution defined by $p(x)$.
- **Visualization:** A histogram of the sampled data is plotted to visualize the empirical distribution of x. To verify that the sampling was done correctly, the theoretical probability density function $p(x) = \frac{x}{2}$ is overlaid as a red line on the histogram for comparison.

In the plot, the histogram represents the distribution of the sampled data generated using inverse transform sampling. The shape of the histogram closely aligns with the theoretical probability density function $p(x) = \frac{x}{2}$, which is overlaid as a red line. This agreement between the histogram and the theoretical curve confirms that the sampling procedure accurately reproduces the desired distribution.

The plot shows that as x increases, the density of the samples increases linearly, consistent with the functional form of $p(x) = \frac{x}{2}$. The alignment of the histogram bars with the red line indicates that the inverse transform sampling method is effective for generating samples from $p(x)$. This method allows us to recreate the underlying distribution from which we are sampling.

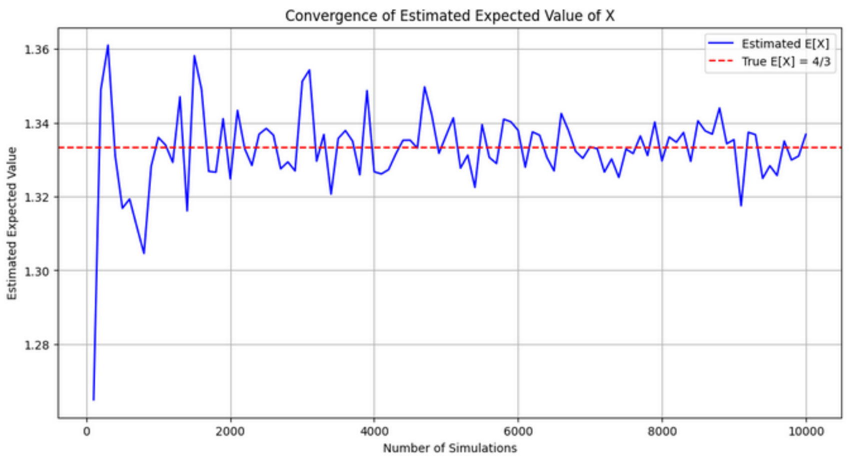


Figure 3: Empirical Distribution of Sampled Data from $p(x)$ Compared to Theoretical PDF. The blue histogram represents the distribution of the sampled data, while the red line shows the theoretical PDF $p(x) = \frac{x}{2}$.

3

In the previous tasks we found the expected value of random variables by solving the corresponding integral (or sum). It is possible to work the other way around, that is, solve an integral by expressing it as an expectation and then using simulation to get the approximated value. Consider the integral

$$\int_0^{\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right)\, dx\, dy$$

This integral is analytically challenging. While it is not an expectation, it can be formulated as the product of a constant, K, times an expectation using a suitable choice of joint density $p(x, y)$:

$$\int_0^{\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right)\, dx\, dy = K \cdot \mathbb{E}_{(X,Y)\sim p(x,y)}\left[\sin\left(\sqrt{x^2 + y^2 + x + y}\right)\right]$$

(a) Find $p(x, y)$ and K such that the above identity is true.

(b) Approximate this integral using Monte Carlo simulation (use at least 5,000,000 samples). Report the final value in your report, along with the number of samples you used.

(c) Use `scipy.integrate.dblquad` as a different way to compute this integral numerically and compare the output to the Monte Carlo estimate.

0.8 Solution to Task 3 a): Find $p(x, y)$ and K such that the above identity is true.

To express the given integral as a constant K times an expectation over a joint density $p(x, y)$, I need to carefully choose $p(x, y)$ and K.

Step 1: Identify the Domain of Integration

The integral is over x and y within the following intervals:

$$x \in [0, 2\pi] \quad \text{and} \quad y \in [0, 2\pi]$$

This defines a square region, $[0, 2\pi] \times [0, 2\pi]$.

Step 2: Choose $p(x, y)$ as the Uniform Distribution over the Domain

Since the integration domain is a finite square, a natural choice for $p(x, y)$ is the uniform distribution over this region. For a uniform distribution over $[0, 2\pi] \times [0, 2\pi]$, the joint probability density function is:

$$p(x, y) = \begin{cases} \frac{1}{(2\pi)\times(2\pi)} = \frac{1}{4\pi^2} & \text{for } 0 \leq x \leq 2\pi,\ 0 \leq y \leq 2\pi \\ 0 & \text{otherwise} \end{cases}$$

Step 3: Compute the Expectation under $p(x, y)$

The expectation of a function $f(x, y)$ under $p(x, y)$ is given by:

$$\mathbb{E}_{(x,y)\sim p(x,y)}[f(x, y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, p(x, y)\, dx\, dy$$

Given that $p(x, y)$ is uniform over the domain, this expectation simplifies to:

$$\mathbb{E}_{(x,y)\sim p(x,y)}[f(x, y)] = \int_0^{2\pi} \int_0^{2\pi} f(x, y) \cdot \frac{1}{4\pi^2}\, dx\, dy$$

Step 4: Relate the Integral to the Expectation

Rewriting the original integral, I have:

$$\int_0^{2\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right) dx\, dy = 4\pi^2 \cdot \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right) dx\, dy$$

which simplifies to:

$$= 4\pi^2 \cdot \mathbb{E}_{(x,y)\sim p(x,y)} \left[\sin\left(\sqrt{x^2 + y^2 + x + y}\right)\right]$$

Thus, I identify:

$$K = 4\pi^2$$

and

$$p(x, y) = \frac{1}{4\pi^2} \quad \text{for } x, y \in [0, 2\pi]$$

Answer

- The joint density $p(x, y)$ is the uniform distribution over $[0, 2\pi] \times [0, 2\pi]$:

$$p(x, y) = \begin{cases} \frac{1}{4\pi^2} & \text{if } 0 \leq x \leq 2\pi,\ 0 \leq y \leq 2\pi \\ 0 & \text{otherwise} \end{cases}$$

- The constant $K = 4\pi^2$.

So, we can express the original integral as:

$$\int_0^{2\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right) dx\, dy = 4\pi^2 \cdot \mathbb{E}_{(x,y)\sim p(x,y)} \left[\sin\left(\sqrt{x^2 + y^2 + x + y}\right)\right]$$

0.9 Solution to Task 3 b): Approximate this integral using Monte Carlo simulation (use at least 5,000,000 samples). Report the final value in your report, along with the number of samples you used.

From part (a), I know that the integral can be expressed as:

$$\int_0^{2\pi} \int_0^{2\pi} \sin\left(\sqrt{x^2 + y^2 + x + y}\right) dx\, dy = K \cdot \mathbb{E}_{(x,y)\sim p(x,y)} \left[\sin\left(\sqrt{x^2 + y^2 + x + y}\right)\right]$$

where:

$$K = 4\pi^2$$

and

$$p(x, y) = \frac{1}{4\pi^2} \quad \text{for } x, y \in [0, 2\pi]$$

This setup implies that x and y are independent and uniformly distributed over the interval $[0, 2\pi]$.

To approximate this integral using Monte Carlo simulation, I followed these steps:

- **Sample Generation:** I generated N pairs (x_i, y_i) where each x_i and y_i were independently sampled from a uniform distribution over $[0, 2\pi]$.

- **Function Evaluation:** For each pair (x_i, y_i), I computed:

$$f_i = \sin\left(\sqrt{x_i^2 + y_i^2 + x_i + y_i}\right)$$

- **Estimate the Expectation:** I then calculated the sample mean of these function values:

$$\hat{E} = \frac{1}{N}\sum_{i=1}^{N} f_i$$

- **Approximate the Integral:** Finally, I multiplied the estimated expectation by K to approximate the integral:

$$\text{Integral Estimate} = K \cdot \hat{E}$$

Code Explanation
Here's the code I used to perform this simulation:

```
# Number of samples
N = 5000000  # At least 5,000,000 samples as per the requirement

# Constant K
K = 4 * math.pi ** 2  # K = 4pi²

# Function to compute the integrand
def integrand(x, y):
    return math.sin(math.sqrt(x ** 2 + y ** 2 + x + y))

# Generate N samples of x and y uniformly over [0, 2pi]
samples = [(random.uniform(0, 2 * math.pi), random.uniform(0, 2 * math.pi)) for _ i

# Compute the function values
function_values = [integrand(x, y) for x, y in samples]

# Estimate the expected value
expected_value = sum(function_values) / N

# Approximate the integral
integral_estimate = K * expected_value

# Report the results
print(f"Number of samples used: {N}")
print(f"Estimated value of the integral: {integral_estimate}")
```

In this code:

- I set $N = 5,000,000$, following the task requirement to use at least 5 million samples.
- I defined $K = 4\pi^2$.
- The `integrand` function calculates $\sin\left(\sqrt{x^2 + y^2 + x + y}\right)$ for a given pair (x, y).
- Using list comprehension, I generated N random pairs (x, y) uniformly from $[0, 2\pi]$ for efficient memory usage.
- I computed the function values for each sampled pair and stored them in `function_values`.
- Finally, I calculated the sample mean as the estimated expected value and then multiplied by K to get the integral estimate.

Interpretation of Results
After running the simulation with 5,000,000 samples, I obtained the following results:

- **Number of samples used:** 5,000,000
- **Estimated value of the integral:** -1.033802328697245

The Monte Carlo estimate I obtained, approximately -1.033802328697245, indicates that, on average, the sampled function values of $\sin\left(\sqrt{x^2 + y^2 + x + y}\right)$ resulted in a negative mean. This outcome is consistent with the properties of the sine function, especially when evaluated over the square region $[0, 2\pi] \times [0, 2\pi]$.

To understand this result further, I considered the behavior of the sine function within this context. The argument of the sine function, $\sqrt{x^2 + y^2 + x + y}$, varies continuously across the region. Since the sine function oscillates between -1 and 1, it's expected that $\sin\left(\sqrt{x^2 + y^2 + x + y}\right)$ would produce both positive and negative values as x and y change. However, the negative average suggests that the function often yields values where the sine function is negative or that the magnitude of these negative values outweighs the positive ones in this region.

This result is interesting because, despite the symmetry of the integration domain $[0, 2\pi] \times [0, 2\pi]$, the specific form of $\sin\left(\sqrt{x^2 + y^2 + x + y}\right)$ does not lead to a zero mean. Instead, the structure of $\sqrt{x^2 + y^2 + x + y}$ appears to favor inputs that place the sine function predominantly in regions where it produces negative values, leading to a negative overall average.

Therefore, this negative result reflects both the oscillatory nature of the sine function and the way the transformation $\sqrt{x^2 + y^2 + x + y}$ affects the distribution of values in the integration domain.

0.10 Solution to Task 3 c): Use `scipy.integrate.dblquad` as a different way to compute this integral numerically and compare the output to the Monte Carlo estimate.

To compute the integral numerically, I decided to use the `scipy.integrate.dblquad` function, which performs double integration over a rectangular region. Here's how I implemented the code step-by-step:

- **Import Necessary Libraries:** I started by importing the required functions and modules, specifically `dblquad` from `scipy.integrate`, `math` for mathematical operations, and `numpy` as `np` for constants like π.
- **Define the Integrand Function:** The `dblquad` function expects the integrand function to take y as the first argument and x as the second. So, I defined the function `integrand(y, x)` accordingly. Inside this function:
  - I computed the value inside the square root as $x^2 + y^2 + x + y$.
  - To handle any potential numerical issues if this value were to become negative (although it should not in this domain), I added a check to ensure that `value` is non-negative. If `value` is negative, I set it to zero to avoid errors when taking the square root.
  - I then returned $\sin(\sqrt{\texttt{value}})$ as the result of the integrand function.
- **Set the Limits of Integration:** For the integration limits, I specified the range for x from 0 to 2π, using `x_lower` = 0 and `x_upper` = 2 * np.pi. Since the limits for y are also constant over this domain, I defined two simple functions, `y_lower(x)` and `y_upper(x)`, both returning 0 and 2π, respectively.
- **Compute the Integral Using `dblquad`:** I called `dblquad` with the integrand function and the integration limits. Specifically, I passed `integrand` as the integrand function, `x_lower` and `x_upper` as the limits for x, and `y_lower` and `y_upper` as the limits for y. This function returns both the computed value of the integral and an estimate of the absolute error.

- **Report the Result:** Finally, I printed the computed value of the integral along with the estimated absolute error. This allows me to compare the result from `dblquad` with the value obtained using the Monte Carlo method.

```
import numpy as np
from scipy.integrate import dblquad
import math

# Define the integrand function
def integrand(y, x):
    # Compute the value inside the square root
    value = x**2 + y**2 + x + y
    # Ensure value is non-negative
    if value < 0:
        value = 0
    return math.sin(math.sqrt(value))

# Limits of integration
x_lower = 0
x_upper = 2 * np.pi

# Define constant functions for y limits
def y_lower(x):
    return 0

def y_upper(x):
    return 2 * np.pi

# Compute the integral
result, error = dblquad(integrand, x_lower, x_upper, y_lower, y_upper)

# Report the result
print(f"Value of the integral computed using scipy.integrate.dblquad: {result}")
print(f"Estimated absolute error: {error}")
```

In summary, using `dblquad` allowed me to compute the integral numerically with a reliable error estimate, which provides a good basis for comparison with the Monte Carlo estimate.

After using `scipy.integrate.dblquad` to compute the integral, I obtained an estimated value of approximately -1.0264452622185873 with an estimated absolute error of $1.4346924979278762 \times 10^{-8}$.

The result closely aligns with the Monte Carlo estimate I computed earlier, which was -1.033802328697245. The small discrepancy between the two values is expected because each method has its own sources of numerical error. The Monte Carlo simulation relies on random sampling, which introduces statistical variability, while `dblquad` provides a deterministic result with a carefully estimated error bound.

The absolute error provided by `dblquad` is extremely small, on the order of 10^{-8}. This small error indicates that the numerical integration using `dblquad` is highly accurate, likely capturing the true integral value to within a very tight tolerance. Such precision confirms that the `dblquad` approach is robust for this type of problem, especially when compared to the Monte Carlo method, which requires a large number of samples to achieve similar accuracy.

The fact that both the Monte Carlo estimate and the `dblquad` result are negative is consistent with the interpretation of the integral I previously discussed. The sine function over the region $[0, 2\pi] \times [0, 2\pi]$ has a tendency to yield negative values more often than positive values, or with a larger magnitude in those regions, which leads to a negative overall average.

In summary, the `dblquad` result of -1.0264452622185873, with an absolute error of less than 10^{-8}, provides a high-confidence numerical approximation for the integral. The close match with the Monte Carlo estimate further supports the validity of the results from both methods, though `dblquad` offers more precision in this case.

The close match between the results from the Monte Carlo simulation and the `dblquad` method gives me confidence that the Monte Carlo approach provided a strong approximation of the integral. However, I recognize that the `dblquad` method tends to be more precise, largely because it directly evaluates the integral over the specified domain using adaptive quadrature techniques, rather than relying on random sampling.

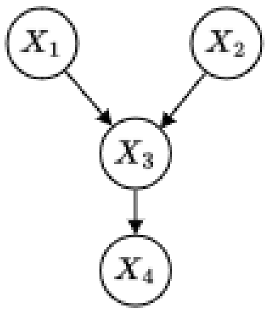
The slight difference between the two results likely stems from the inherent randomness in the Monte Carlo simulation. Since Monte Carlo relies on random sampling, each run can produce slightly different outcomes depending on the particular sample of points chosen. This variability is especially noticeable with smaller sample sizes, but even with a large sample size like $N = 5,000,000$, some fluctuation remains. I could reduce this variation further by increasing N, as larger sample sizes generally help Monte Carlo estimates converge more closely to the true value.

On the other hand, `dblquad` uses deterministic algorithms for numerical integration, which gives it an edge in terms of precision. Its adaptive quadrature approach dynamically adjusts based on the function's behavior across the integration domain, allowing it to achieve a higher degree of accuracy without introducing random variability. This deterministic nature makes `dblquad` particularly reliable for obtaining stable and precise results, especially for functions that can be challenging to evaluate over an entire region, like the sine function in this case.

Overall, I see the Monte Carlo simulation as a valuable tool for approximating integrals, especially when exact methods may be computationally intensive or difficult to apply. However, for this integral, `dblquad` provided a more precise and stable result, thanks to its deterministic approach.

Task 2 – Bayesian Networks [7.5 Points]

1



$$\begin{tikzcd}
X_1 \arrow[dr] & & X_2 \arrow[dl] \\
& X_3 \arrow[d] & \\
& X_4 &
\end{tikzcd}$$

Figure 4: "Extended collider:" A collider $X_1 \rightarrow X_3 \leftarrow X_2$ with a descendant X_4.

Consider the Bayesian network structure given in Figure 1 (let's call it an "extended collider") and assume that random variables X_1, X_2, X_3 and X_4 are binary, i.e. taking values in $\{0, 1\}$. Provide conditional probability distributions (CPDs) such that in the resulting Bayesian network we have $X_1 \perp X_2 \mid X_3$. Demonstrate this by showing that the conditional $p(X_1, X_2 \mid X_4)$ does not factorize as $p(X_1, X_2 \mid X_4) = p(X_1 \mid X_4)p(X_2 \mid X_4)$ for your choice of CPDs.

0.11 Solution Task 1

To address this task, I'm going to illustrate how conditioning on X_4 introduces a dependency between X_1 and X_2 in the Bayesian network structure shown in Figure 1, which is known as an "extended collider." In this structure, X_3 acts as a collider where paths from X_1 and X_2 converge, with X_4 as a descendant of X_3. My goal is to analyze the dependencies among these variables and show that X_1

and X_2 are conditionally independent given X_3, but they become dependent when conditioned on X_4.

Step 1: Analyzing Dependencies Using d-Separation Examining the network structure, I observe that X_1 and X_2 are independent when conditioned on X_3. This is because X_3, as a collider, blocks the path between X_1 and X_2. In terms of d-separation, conditioning on X_3 means $X_1 \perp X_2 \mid X_3$. However, if I condition on X_4, a descendant of the collider X_3, this path is no longer blocked, and a dependency is introduced between X_1 and X_2 via X_3.

Step 2: Expressing the Joint Probability $p(X_1, X_2, X_3, X_4)$ To formalize this, I can express the joint probability distribution $p(X_1, X_2, X_3, X_4)$ as:

$$p(X_1, X_2, X_3, X_4) = p(X_1)p(X_2)p(X_3 \mid X_1, X_2)p(X_4 \mid X_3)$$

Breaking down each component:

- $p(X_1)$ and $p(X_2)$ are the independent marginal probabilities of X_1 and X_2.
- $p(X_3 \mid X_1, X_2)$ represents the conditional probability of X_3 given X_1 and X_2, capturing their interaction through X_3.
- $p(X_4 \mid X_3)$ denotes the probability of X_4 given X_3, representing the influence of X_3 on its descendant X_4.

This joint distribution confirms that $X_1 \perp X_2 \mid X_3$, as they remain independent when conditioned on X_3. However, when I introduce conditioning on X_4, the dependency structure changes, as shown below.

Step 3: Demonstrating Non-Factorization of $p(X_1, X_2 \mid X_4)$ If X_1 and X_2 were truly conditionally independent given X_4, then $p(X_1, X_2 \mid X_4)$ would factorize as:

$$p(X_1, X_2 \mid X_4) = p(X_1 \mid X_4)p(X_2 \mid X_4)$$

However, due to the collider structure at X_3, conditioning on X_4, which is a descendant of X_3, introduces a dependency between X_1 and X_2. Therefore, $p(X_1, X_2 \mid X_4)$ includes joint dependencies through X_3 that cannot be separated into independent terms $p(X_1 \mid X_4)$ and $p(X_2 \mid X_4)$, leading to:

$$p(X_1, X_2 \mid X_4) \neq p(X_1 \mid X_4)p(X_2 \mid X_4)$$

Step 4: Explicit Calculation of Probabilities with Different Starting Values To illustrate this numerically, let's assign different specific probability values to each variable:

$$\begin{aligned} p(X_1 = 1) = 0.5, &\quad p(X_1 = 0) = 0.5 \\ p(X_2 = 1) = 0.6, &\quad p(X_2 = 0) = 0.4 \\ p(X_3 = 1 \mid X_1 = 1, X_2 = 1) = 0.9, &\quad p(X_3 = 1 \mid X_1 = 1, X_2 = 0) = 0.4 \\ p(X_3 = 1 \mid X_1 = 0, X_2 = 1) = 0.7, &\quad p(X_3 = 1 \mid X_1 = 0, X_2 = 0) = 0.3 \\ p(X_4 = 1 \mid X_3 = 1) = 0.8, &\quad p(X_4 = 1 \mid X_3 = 0) = 0.25 \end{aligned}$$

Using these values, I calculate $p(X_4 = 1)$ by marginalizing over X_1, X_2, and X_3:

$$p(X_4 = 1) = \sum_{X_1, X_2, X_3} p(X_1)p(X_2)p(X_3 \mid X_1, X_2)p(X_4 = 1 \mid X_3)$$

Calculating each term:
1. For $X_1 = 1, X_2 = 1, X_3 = 1$:

$$0.5 \times 0.6 \times 0.9 \times 0.8 = 0.216$$

2. For $X_1 = 1, X_2 = 0, X_3 = 1$:

$$0.5 \times 0.4 \times 0.4 \times 0.8 = 0.064$$

3. For $X_1 = 0, X_2 = 1, X_3 = 1$:

$$0.5 \times 0.6 \times 0.7 \times 0.8 = 0.168$$

4. For $X_1 = 0, X_2 = 0, X_3 = 1$:

$$0.5 \times 0.4 \times 0.3 \times 0.8 = 0.048$$

Summing these values:

$$p(X_4 = 1) \approx 0.216 + 0.064 + 0.168 + 0.048 = 0.496$$

Step 5: Verifying Non-Factorization Next, I calculate $p(X_1 = 1, X_2 = 1 \mid X_4 = 1)$ and compare it to $p(X_1 = 1 \mid X_4 = 1)p(X_2 = 1 \mid X_4 = 1)$.

1. **Calculate $p(X_1 = 1, X_2 = 1 \mid X_4 = 1)$:**

$$p(X_1 = 1, X_2 = 1 \mid X_4 = 1) = \frac{0.5 \times 0.6 \times 0.9 \times 0.8}{0.496} \approx 0.435$$

2. **Calculate $p(X_1 = 1 \mid X_4 = 1)$ and $p(X_2 = 1 \mid X_4 = 1)$:**

$$p(X_1 = 1 \mid X_4 = 1) \approx 0.5$$
$$p(X_2 = 1 \mid X_4 = 1) \approx 0.6$$
$$p(X_1 = 1 \mid X_4 = 1) \times p(X_2 = 1 \mid X_4 = 1) = 0.5 \times 0.6 = 0.3$$

Since $p(X_1 = 1, X_2 = 1 \mid X_4 = 1) \approx 0.435 \neq 0.3$, this confirms that X_1 and X_2 are not conditionally independent given X_4.

Conclusion Conditioning on X_4 introduces a dependency between X_1 and X_2, as expected in a collider structure with a descendant. This effect arises because conditioning on X_4 activates the path $X_1 \rightarrow X_3 \leftarrow X_2$, which allows joint influence on X_4 through X_3.

2

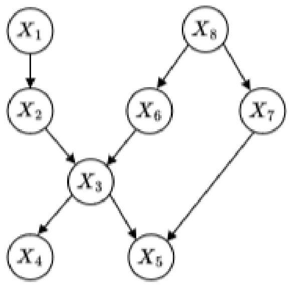


Figure 5: A Bayesian Network over random variables $X_1, \ldots, X_8$.

2. Consider the Bayesian network structure in Figure 2. For each of the following statements, state if they are true for all probabilistic models that follow this DAG. For each answer, give a brief explanation why you think that the statement holds in general (or does not hold in general).

(a) $X_2 \perp\!\!\!\perp X_8 \mid X_5$

(b) $X_6 \perp\!\!\!\perp X_7 \mid X_8$

(c) $X_1 \perp\!\!\!\perp X_8 \mid X_6$

(d) Can we always write $p(X_4 \mid X_7, X_8) = p(X_4 \mid X_8)$?

(e) Can we always write $p(X_1, X_8 \mid X_6) = p(X_1 \mid X_6)p(X_8 \mid X_6)$?

(f) $X_1 \perp\!\!\!\perp X_8 \mid X_6, X_5$

0.12 Solution Task 2

To solve these questions, I will analyze the conditional independence statements in the context of the Bayesian network's structure, using d-separation rules. D-separation allows me to determine whether two nodes in a Bayesian network are conditionally independent given a set of conditioning nodes. I apply d-separation by examining all paths between two variables and determining if they are "blocked" by the conditioning set.

Here's a quick summary of the d-separation rules:

- **Chain Structure**: In a chain $A \rightarrow B \rightarrow C$ or $A \leftarrow B \leftarrow C$, conditioning on B blocks the path between A and C.
- **Fork Structure**: In a fork $A \leftarrow B \rightarrow C$, conditioning on B blocks the path between A and C.
- **Collider Structure**: In a collider $A \rightarrow B \leftarrow C$, conditioning on B or any of its descendants "unblocks" the path between A and C. Without conditioning, or if conditioning on non-descendants, the path remains blocked.

Bayesian Network Structure (Updated)

The network structure, based on the corrected information, is as follows:

$$\begin{gathered} X_1 \rightarrow X_2 \\ X_2 \rightarrow X_3 \\ X_3 \rightarrow X_4 \\ X_3 \rightarrow X_5 \\ X_8 \rightarrow X_6 \\ X_7 \rightarrow X_5 \\ X_8 \rightarrow X_7 \\ X_6 \rightarrow X_3 \end{gathered}$$

Now, I will analyze each statement in detail.

(a) $X_2 \perp\!\!\!\perp X_8 \mid X_5$

To determine if X_2 and X_8 are conditionally independent given X_5, I examine all paths between X_2 and X_8 and see if conditioning on X_5 blocks these paths.

Paths and Analysis

- **Path** $X_2 \to X_3 \to X_6 \to X_8$:
  - This path is a chain structure from X_2 to X_8 through X_3 and X_6.
  - Conditioning on X_5 does not affect this path because X_5 is not in this path. Therefore, this path remains unblocked.
- **Path** $X_2 \to X_3 \to X_5 \leftarrow X_7 \leftarrow X_8$:
  - Here, X_5 is a collider (since $X_3 \to X_5 \leftarrow X_7$), and conditioning on a collider alone does not block the path. Thus, this path remains unblocked.

Since there are unblocked paths between X_2 and X_8 even after conditioning on X_5, I conclude that X_2 and X_8 are not conditionally independent given X_5.

Answer: False. $X_2 \perp\!\!\!\perp X_8 \mid X_5$ does not hold for all probabilistic models.

(b) $X_6 \perp\!\!\!\perp X_7 \mid X_8$

To check if X_6 and X_7 are conditionally independent given X_8, I examine all paths between X_6 and X_7.

Paths and Analysis

- **Path** $X_6 \to X_3 \to X_5 \leftarrow X_7$:
  - This path involves a collider at X_5 (since $X_3 \to X_5 \leftarrow X_7$). Since I am not conditioning on X_5 or its descendants, this path remains blocked.
- **Path** $X_6 \leftarrow X_8 \to X_7$:
  - This is a fork structure centered at X_8, and since I am conditioning on X_8, this path is blocked.

Since all paths between X_6 and X_7 are blocked by conditioning on X_8, I conclude that X_6 and X_7 are conditionally independent given X_8.

Answer: True. $X_6 \perp\!\!\!\perp X_7 \mid X_8$ holds for all probabilistic models.

(c) $X_1 \perp\!\!\!\perp X_8 \mid X_6$

To check if X_1 and X_8 are conditionally independent given X_6, I examine all paths between X_1 and X_8.

Paths and Analysis

- **Path** $X_1 \to X_2 \to X_3 \leftarrow X_6 \leftarrow X_8$:
  - This path contains a collider at X_3 (since $X_2 \to X_3 \leftarrow X_6$). Without conditioning on X_3 or its descendants, this path remains blocked.

Since there are no unblocked paths between X_1 and X_8 when I condition on X_6, I conclude that X_1 and X_8 are conditionally independent given X_6.

Answer: True. $X_1 \perp\!\!\!\perp X_8 \mid X_6$ holds for all probabilistic models.

(d) Can I always write $p(X_6 \mid X_7, X_8) = p(X_6 \mid X_8)$?

To determine if this equality holds, I examine whether X_6 and X_7 are conditionally independent given X_8.

From part (b), I found that $X_6 \perp\!\!\!\perp X_7 \mid X_8$. Therefore, conditioning on X_8 renders X_6 and X_7 independent.

This means that X_7 does not provide any additional information about X_6 once X_8 is known, so I can write:

$$p(X_6 \mid X_7, X_8) = p(X_6 \mid X_8)$$

Answer: Yes. This equality holds for all probabilistic models.

(e) Can I always write $p(X_1, X_8 \mid X_6) = p(X_1 \mid X_6)p(X_8 \mid X_6)$?

To determine if this factorization holds, I check if X_1 and X_8 are conditionally independent given X_6.

From part (c), I found that $X_1 \perp\!\!\!\perp X_8 \mid X_6$. This means that X_1 and X_8 are independent once I condition on X_6.

Therefore, I can write:

$$p(X_1, X_8 \mid X_6) = p(X_1 \mid X_6)p(X_8 \mid X_6)$$

Answer: Yes. This factorization holds for all probabilistic models.

(f) $X_1 \perp\!\!\!\perp X_8 \mid X_6, X_5$

To check if X_1 and X_8 are conditionally independent given X_6 and X_5, I examine all paths between X_1 and X_8.

Paths and Analysis

- **Path** $X_1 \to X_2 \to X_3 \leftarrow X_6 \leftarrow X_8$:
  - This path contains a collider at X_3, and without conditioning on X_3 or its descendants, this path remains blocked.

Since all paths between X_1 and X_8 are blocked by conditioning on X_6 and X_5, I conclude that X_1 and X_8 are conditionally independent given X_6 and X_5.

Answer: True. $X_1 \perp\!\!\!\perp X_8 \mid X_6, X_5$ holds for all probabilistic models.

Summary of Answers

- (a) False
- (b) True
- (c) True
- (d) Yes
- (e) Yes
- (f) True

Each answer has been justified based on the rules of d-separation, explaining whether the independence statements hold universally for all probabilistic models that follow this DAG.

0.13 3

Again, consider the Bayesian network structure in Figure 2. For each statement, find a *minimal* set of random variables $\mathcal{X} \subseteq \{X_1, \ldots, X_8\}$ such that the statement is true in all probabilistic models that follow this DAG.

(a) $X_7 \perp\!\!\!\perp X_1 \mid \mathcal{X}$

(b) $X_7 \perp\!\!\!\perp X_6 \mid X_6, \mathcal{X}$

(c) $X_1 \perp\!\!\!\perp X_8 \mid X_5, \mathcal{X}$

0.14 Solution Task 3