IDS 576

Team 3

Assignment 4

Rahul Shukla (652959493)
Scott Brewer (676075252)

Assignment 4

Question1: -

Joint distribution of A, B, C, D

A	B	C	D	P(A,B,C,D)
0	0	1	10	P(A=0,B=0,C=1,D=10)
0	0	1	20	P(A=0,B=0,C=1,D=20)
0	0	2	10	P(A=0,B=0,C=2,D=10)
0	0	2	20	P(A=0,B=0,C=2,D=20)
0	0	3	10	P(A=0,B=0,C=3,D=10)
0	0	3	20	P(A=0,B=0,C=3,D=20)
0	1	1	10	P(A=0,B=1,C=1,D=10)
0	1	1	20	P(A=0,B=1,C=1,D=20)
0	1	2	10	P(A=0,B=1,C=2,D=10)
0	1	2	20	P(A=0,B=1,C=2,D=20)
0	1	3	10	P(A=0,B=1,C=3,D=10)
0	1	3	20	P(A=0,B=1,C=3,D=20)
1	0	1	10	P(A=1,B=0,C=1,D=10)
1	0	1	20	P(A=1,B=0,C=1,D=20)
1	0	2	10	P(A=1,B=0,C=2,D=10)
1	0	2	20	P(A=1,B=0,C=2,D=20)
1	0	3	10	P(A=1,B=0,C=3,D=10)
1	0	3	20	P(A=1,B=0,C=3,D=20)
1	1	1	10	P(A=1,B=1,C=1,D=10)
1	1	1	20	P(A=1,B=1,C=1,D=20)
1	1	2	10	P(A=1,B=1,C=2,D=10)
1	1	2	20	P(A=1,B=1,C=2,D=20)
1	1	3	10	P(A=1,B=1,C=3,D=10)
1	1	3	20	P(A=1,B=1,C=3,D=20)
2	0	1	10	P(A=2,B=0,C=1,D=10)
2	0	1	20	P(A=2,B=0,C=1,D=20)
2	0	2	10	P(A=2,B=0,C=2,D=10)
2	0	2	20	P(A=2,B=0,C=2,D=20)
2	0	3	10	P(A=2,B=0,C=3,D=10)
2	0	3	20	P(A=2,B=0,C=3,D=20)
2	1	1	10	P(A=2,B=1,C=1,D=10)
2	1	1	20	P(A=2,B=1,C=1,D=20)
2	1	2	10	P(A=2,B=1,C=2,D=10)
2	1	2	20	P(A=2,B=1,C=2,D=20)
2	1	3	10	P(A=2,B=1,C=3,D=10)
2	1	3	20	P(A=2,B=1,C=3,D=20)

Values required for joint distribution with A, B, C, D = (3*2*3*2) -1 = 35

Possible factorization for A, B, C: - P (A, B, C)

A	B	C
0	0	1
0	0	2
0	0	3
0	1	1
0	1	2
0	1	3
1	0	1
1	0	2
1	0	3
1	1	1

1	1	2
1	1	3
2	0	1
2	0	2
2	0	3
2	1	1
2	1	2
2	1	3

Factorization possible for P (A, B)

A	B
0	0
0	1
1	0
1	1
2	0
2	1

2. Because P (A|B, C, D) = P (A|B), that's why A doesn't depends on C and D

Because P (C|D) = P (C), that's why C doesn't depends on D

Hence, P (A|B, C, D) = P (A|B) * P (B|C, D) * P (C) *P (D)

CPD of P (A|B), no. of values = (2-1)*3 = 3

CPD of P (B|C, D), no. of values = (61)*2 = 10

PD of P (C), no. of values = (3-1) = 2

PD of P (D), no. of values = (2-1) = 1

Total values = 3+10+2+1 = 16

3. P (C|B, D) = P (C|B), means C doesn't depends on D. DAG also means C depends on B but not D.

That's why, P (C|B, D) = P (C|B) is valid

4. P(A,B,C,D) = P(A) P(B) P(C) P(D), shows A,B,C,D are independent of one another

PD of P(C), no. of values = (3-1) = 2

PD of P(C), no. of values = (2-1) = 1

PD of P(C), no. of values = (3-1) = 2

PD of P(C), no. of values = (2-1) = 1

Total values = 2+1+2+1 = 6

- CPD = Conditional Probability Distribution
- PD= Probability Distribution

Q

Given $Z_4 = X_1 \oplus X_2$ and $Z_5 = X_2 \oplus X_3$

2.1 $P(X_2, X_3 | Z_5 = 0)$ and $P(X_2, X_3 | Z_5 = 1) = ?$

from XOR relationship, we know

$$P(Z_5 = 1) = P(X_2 = 0, X_3 = 1) + P(X_2 = 1, X_3 = 0) \rightarrow (1)$$
$$P(Z_5 = 0) = P(X_2 = 0, X_3 = 0) + P(X_2 = 1, X_3 = 1) \rightarrow (2)$$

(a) $$P(X_2 = 0, X_3 = 0 | Z_5 = 0) = \frac{P(Z_5 = 0 | X_2 = 0, X_3 = 0) P(X_2 = 0 | X_3 = 0) P(X_3 = 0)}{P(Z_5 = 0)}$$

Since X_2 and X_3 are independent

$$P(X_2 = 0 | X_3 = 0) = P(X_2 = 0)$$

Hence

$$P(X_2 = 0, X_3 = 0 | Z_5 = 0) = \frac{P(Z_5 = 0 | X_2 = 0, X_3 = 0) P(X_2 = 0) P(X_3 = 0)}{P(Z_5 = 0)}$$

$$= \frac{1 \cdot (1-p)(1-p)}{(1-p)(1-p) + p \cdot p} \rightarrow \text{from } (2)$$

$$= \frac{(1-p)^2}{(1-p)^2 + p^2}$$

Similarly

$$P(X_2 = 1, X_3 = 1 | Z_5 = 0) = \boxed{\frac{p^2}{p^2 + (1-p)^2}}$$

$$P(X_2=1, X_3=0 \mid Z_5=0) = \frac{P(Z_5=0 \mid X_2=1, X_3=0)\,P(X_2=1)\,P(X_3=0)}{P(Z_5=0)}$$

$$= \frac{0 \cdot p \cdot (1-p)}{(1-p)^2 + p^2} = 0$$

$$P(X_2=0, X_3=1 \mid Z_5=0) = \frac{P(Z_5=0 \mid X_2=0, X_3=1)\,P(X_2=0)\,P(X_3=1)}{P(Z_5=0)}$$

$$= \frac{0 \cdot (1-p) \cdot p}{(1-p)^2 + p^2} = 0$$

$$P(X_2=0, X_3=0 \mid Z_5=1) = \frac{P(Z_5=1 \mid X_2=0, X_3=0)\,P(X_2=0)\,P(X_3=0)}{P(Z_5=1)}$$

$$= \frac{0 \cdot (1-p)(1-p)}{p(1-p)+(1-p)p} = 0$$

$$P(X_2=1, X_3=1 \mid Z_5=1) = \frac{P(Z_5=1 \mid X_2=1, X_3=1)\,P(X_2=1)\,P(X_3=1)}{P(Z_5=1)}$$

$$= \frac{0 \cdot p \cdot p}{p(1-p)+p(1-p)} = 0$$

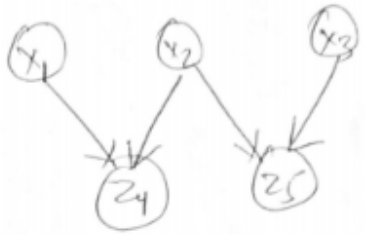
$$P(X_2=1, X_3=0 \mid Z_5=1) = \frac{P(Z_5=1 \mid X_2=1, X_3=0)\,P(X_2=1)\,P(X_3=0)}{P(Z_5=1)}$$

$$= \frac{1 \cdot p \cdot (1-p)}{p(1-p)+p(1-p)} = \frac{p(1-p)}{2p(1-p)} = 1/2$$

$$P(X_2=0, X_3=1 \mid Z_5=1) = \frac{P(Z_5=1 \mid Y_2=0, X_3=1)\, P(X_2=0)\, P(X_3=1)}{P(Z_5=1)}$$

$$= \frac{1 \cdot (1-p) \cdot p}{2p(1-p)} = 1/2$$

(22) DPGM →



we know

probability for $\{X_1 = X_2 = X_3 = 1\} \Rightarrow p$

Probability for $\{X_1 = X_2 = X_3 = 0\} \Rightarrow (1-p)$

Conditional Probability for, $Z_4 = X_1 \oplus X_2$

X_1	X_2	$Z_4=1$	$Z_4=0$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Conditional Probability for $Z_5 = X_2 \oplus X_3$.

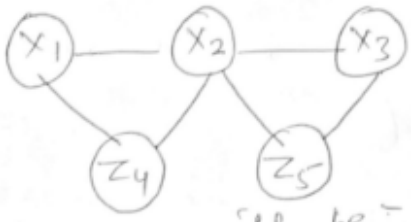
X_2	X_3	$Z_5=1$	$Z_5=0$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The conditional independence relations captured by DPGM are

$X_1 \perp (X_2, X_3, Z_5)$
$X_1 \perp X_2 \mid (X_3, Z_5)$
$X_1 \perp Z_5 \mid (X_2, X_3, Z_4)$
$X_1 \perp (X_2, X_3) \mid Z_5$
$X_1 \perp (X_2, Z_5) \mid X_3$
$X_1 \perp (X_3, Z_5) \mid (X_2, Z_4)$
$X_2 \perp (X_1, X_3)$
$X_2 \perp X_3 \mid (X_1, Z_4)$
$X_3 \perp (X_1, X_2, Z_4)$
$X_3 \perp (X_1, X_2) \mid Z_4$
$X_3 \perp (X_2, Z_4) \mid X_1$
$X_3 \perp (X_1, Z_4) \mid (X_2, Z_5)$
$X_3 \perp Z_4 \mid (X_1, X_2, Z_5)$

$Z_4 \perp (X_3, Z_5) \mid (X_1, X_2)$
$Z_4 \perp Z_5 \mid (X_1, X_2, X_3)$
$Z_5 \perp (X_1, Z_4) \mid (X_2, X_3)$
$(X_1, Z_4) \perp (X_3, Z_5) \mid X_2$

(2.3) UPGM



The corresponding factors will be -

$$\psi_{\{X_1, Z_4, X_2\}}(X_1, Z_4, X_3) = P(X_1)P(X_2)P(Z_4 \mid X_1, X_2)$$

$$\psi_{\{X_2, Z_5, X_3\}}(X_2, Z_5, X_3) = P(X_3)P(Z_5 \mid X_2, X_3)$$

Independent relations captured by the UPGM are -

- $(X_1, Z_4) \perp (X_3, Z_5) \mid X_2$
- $Z_4 \perp (X_3, Z_5) \mid (X_2, X_2)$
- $(X_1, Z_4) \perp Z_5 \mid (X_2, X_3)$
- $X_1 \perp (X_3, Z_5) \mid (Z_4, X_2)$
- $(X_1, Z_4) \perp X_3 \mid (X_3, Z_5)$
- $Z_4 \perp Z_5 \mid (X_1, X_2, X_3)$
- $Z_4 \perp X_3 \mid (X_1, X_2, Z_5)$
- $X_1 \perp Z_5 \mid (Z_4, X_2, X_3)$
- $X_1 \perp X_3 \mid (Z_4, X_2, Z_5)$

(2.4) X_i will be a constant random variable when $P\{0,1\}$ and $i = \{1,2,3\}$, which means Z_4 and Z_5 will be constant as well

Hence $Z_4 \perp X_i$ and $Z_5 \perp X_3$.

if $0 < P < 1$, for independence, the condition would be:

$$P(Z_5=1) = P(Z_5=1 \mid X_3=0) = P(Z_5=1 \mid X_3=1)$$

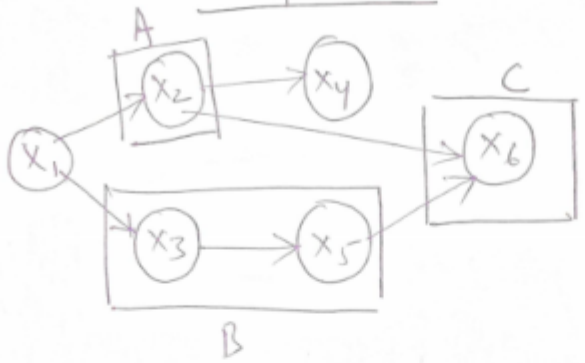
$$P(Z_5=1 \mid X_3=0) = P(X_2=1) = P$$

$$P(Z_5=1 \mid X_3=1) = P(X_2=0) = 1-P$$

$$P(Z_5=1) = P(Z_5=1 \mid X_3=0)\,P(X_3=0) + P(Z_5=1 \mid X_3=1)\,P(X_3=1) = 2P(1-P)$$

so for independence of $0 < P < 1$, P can only be $P = 1/2$ which is not implied by the DPGM and UPGM

Ans 3. D- Separation



Given $A = \{X_2\}$, $B = \{X_3, X_5\}$ $C = \{X_1, X_6\}$.

To prove $A \perp B \mid C$, we need to show if A and B are d-separated by C.

A and B are d-separated by C if all paths from a vertex of A to a vertex of B are blocked w.r.t. C.

assuming $X_i \perp X_j \mid X_k \Rightarrow A \perp B \mid C$

for i and j to be d-separated by k

This is a head-head relationship.

$\{\rightarrow \circ \leftarrow\}$ blocked if $v \notin C$ and no descendents $\in C$.

But in our case the condition is not satisfied

Hence, $A \perp B \mid C$ does not hold!

Q.4 Given: $P(G_1) = [1/2, 1/2]$

$P(G_i = G_1 | G_1) = 0.8 \quad i \in \{2, 3\}$

$P(x_i | G_i = 1) = \mathcal{N}(x_i | \mu = 100, \sigma^2 = 10)$

$P(x_i | G_i = 2) = \mathcal{N}(x_i | \mu = 10, \sigma^2 = 20)$

if $G_i = 1$ - the component is running

and if $G_i = 2$ - Component failed

G_1 is common for both G_2 and G_3 and can influence the performance of G_2 and G_3.

4.1 $X_2 = 100$. Calculate $P(G_1 | X_2 = 100)$

For $G_1 = 1$

$$P(G_1 | X_2 = 100) = \frac{P(X_2 = 100 | G_1 = 1) \; P(G_1 = 1)}{P(X_2 = 100 | G_1 = 1) P(G_1 = 1) + P(X_2 = 100 | G_1 = 2) P(G_1 = 2)} \quad \text{---(1)}$$

Now,

$$P(X_2 = 100 | G_1 = 1) = P(X_2 = 100 | G_2 = 1) P(G_2 = 1 | G_1 = 1) P(G_1) + P(X_2 = 100 | G_2 = 2) P(G_2 = 2 | G_1 = 1) P(G_1) \quad \text{---(2)}$$

$$P(X_2 = 100 | G_1 = 2) = P(X_2 = 100 | G_2 = 1) P(G_2 = 1 | G_1 = 2) P(G_1) + P(X_2 = 100 | G_2 = 2) P(G_2 = 2 | G_1 = 2) P(G_1) \quad \text{---(3)}$$

$$P(X_2 = 100 \mid G_2 = 1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Given, $\mu = 100 \quad \sigma^2 = 10 \quad x = 100$

$$= \frac{1}{\sqrt{10}\sqrt{2\pi}} e^{\left(-\frac{(100-100)^2}{2.10}\right)}$$

$$= \frac{1}{\sqrt{20\pi}} e^0 = 0.126$$

$P(X_2 = 100 \mid G_2 = 2) = 0$

Putting all the value in eqn ①, ② and ③

From eq ②

$P(X_2 = 100 \mid G_1 = 1) = 0.126 * 0.8 * \frac{1}{2} + 0 * 0.2 * \frac{1}{2}$

$= 0.050463$

From eq ③

$P(X_2 = 100 \mid G_1 = 2) = 0.126 * 0.2 * \frac{1}{2} + 0 * 0.8 * \frac{1}{2}$

$= 0.0126$

From eq ①

$$P(G_1 \mid X_2 = 100) = \frac{(0.050463)\left(\frac{1}{2}\right)}{(0.050463)\left(\frac{1}{2}\right) + (0.0126)\left(\frac{1}{2}\right)}$$

$$= \frac{0.050463}{0.050463 + 0.0126} = \underline{0.798}$$

~~4.2~~ $P(G_1 | X_2 = 100)$ For $G_1 = 1$ is,

0.798

$P(G_1 | X_2 = 100)$ For $G_1 = 2$ will be

$\Rightarrow 1 - P(G_1 | X_2 = 100)$

$\Rightarrow 1 - 0.798 = 0.202$

4.2 Calculate $P(G_1 | X_2, X_3)$ For

(a) $X_2 = 100$ and $X_3 = 100$

First we will calculate $P(G_1 | X_2, X_3)$ For $G_1 = 1$

$$P(G_1 | X_2, X_3) = \frac{P(X_2, X_3 | G_1 = 1) P(G_1 = 1)}{P(X_2 X_3 | G_1 = 1) P(G_1 = 1) + P(X_2 X_3 | G_1 = 2) P(G_1 = 2)} \quad \text{---(1)}$$

$$P(X_2; X_3 | G_1 = 1) = P(X_2 | G_1 = 1) * P(X_3 | G_1 = 1) \quad \text{---(2)}$$

$$P(X_2 | G_1 = 1) = P(X_2 | G_2 = 1) P(G_2 = 1 | G_1 = 1) P(G_1) + P(X_2 | G_2 = 2) P(G_2 = 2 | G_1 = 1) P(G_1) \quad \text{---(3)}$$

$$P(X_3 | G_1 = 1) = P(X_3 | G_3 = 1) P(G_3 = 1 | G_1 = 1) P(G_1) + P(X_3 | G_3 = 2) P(G_3 = 2 | G_1 = 1) P(G_1) \quad \text{---(4)}$$

From eq^n (3) and (4) and $X_2 = X_3 = 100$

$$P(X_2 = 100 | G_1 = 1) = \frac{1}{\sqrt{10}\sqrt{2\pi}} e^{\left(\frac{-(100-100)^2}{2.10}\right)} * 0.8 * \frac{1}{2}$$

$$+ 0.0.2 * \frac{1}{2}$$

$$= 0.126 * 0.8 * \frac{1}{2} = 0.050463$$

Similarly,

$$P(X_3 = 100 | G_1 = 1) = 0.050463$$

Putting these value in eq^n (2)

$$P(X_2, X_3 = 100 | G_1 = 1) = 0.050463 * 0.050463$$

Now,

$$P(X_2 X_3 | G_1 = 2) = P(X_2 | G_1 = 2) * P(X_3 | G_1 = 2)$$

$$P(X_2 | G_1 = 2) = P(X_2 | G_2 = 2) P(G_2 = 2 | G_1 = 2) P(G_1)$$
$$+ P(X_2 | G_2 = 1) P(G_2 = 1 | G_1 = 2) P(G_1)$$

$$P(X_3 | G_1 = 2) = P(X_3 | G_3 = 2) P(G_3 = 2 | G_1 = 2) P(G_1)$$
$$+ P(X_3 | G_3 = 1) P(G_3 = 1 | G_1 = 2) P(G_1)$$

$$P(X_2 = 100 | G_1 = 2) = 0 * 0.8 * \frac{1}{2} + 0.126 * 0.2 * \frac{1}{2}$$

$$= 0.0126$$

Similarly, $P(X_3 = 100 | G_1 = 2) = 0.0126$

Putting all the values in eq^n (1)

$$P(G_1|X_2X_3) = \frac{(0.050463 * 0.050463) * \frac{1}{2}}{(0.050463 * 0.050463) * \frac{1}{2} + (0.0126 * 0.0126) * \frac{1}{2}}$$

$$= \frac{0.002546}{0.002546 + 0.000159} = \boxed{0.9413}$$

$P(G_1 | X_2, X_3)$ where $X_2 = X_3 = 100$ and $G_1 = 1$ is $\boxed{0.9413}$

$P(G_1 | X_2, X_3)$ where $G_1 = 2$ will be

$1 - 0.9413 = \boxed{0.058}$

(b) $X_2 = 10$ and $X_3 = 100$

$$P(X_2 = 10 | G_1 = 1) = P(X_2 = 10 | G_2 = 1) P(G_2 = 1 | G_1 = 1) * P(G_1 = 1) + P(X_2 = 10 | G_2 = 2) P(G_2 = 2 | G_1 = 1) P(G_1)$$

$$= 0 * 0.8 * \frac{1}{2} + \frac{1}{\sqrt{20}\sqrt{2\pi}} e^{\left(\frac{-(110-10)^2}{2.20}\right)} * 0.2 * \frac{1}{2}$$

$$= \frac{1}{\sqrt{40\pi}} * 0.2 * \frac{1}{2} = \boxed{0.0089}$$

Similarly, we know $P(X_3 = 100 | G_1 = 1) = 0.050463$

$P(X_2 = 10, X_3 = 100 | G_1 = 1) = 0.0089 * 0.050463$

we know $P(X_3 = 100 | G_1 = 2) = 0.0126$

$P(X_2 = 10 | G_1 = 2) = P(X_2 = 10 | G_2 = 2) P(G_2 = 2 | G_1 = 2) * P(G_1) + P(X_2 = 10 | G_2 = 1) * P(G_2 = 1 | G_1 = 2) P(G_1)$

$= \frac{1}{\sqrt{40\pi}} e^0 * 0.8 * \frac{1}{2} + 0 * 0.2 * \frac{1}{2}$

$= 0.0356$

Hence, $P(G_1 = 1 | X_2 = 10, X_3 = 100) = \frac{0.0126 * (\frac{1}{2})}{(0.0089 * 0.050463) * \frac{1}{2}}$

$\Rightarrow \frac{(0.0089 * 0.050463) * \frac{1}{2}}{(0.0089 * 0.050463) * \frac{1}{2} + (0.0126 * 0.0356) \frac{1}{2}}$

$\Rightarrow \frac{0.00045}{0.00045 + 0.0045} = \boxed{0.499}$

$P(G_1 = 2 | X_2 = 10, X_3 = 100) = 1 - 0.499$

$= \boxed{0.500}$

(c) $X_2 = 10$ and $X_3 = 10$

$P(G_1 = 1 \mid X_2 = 10, X_3 = 10) = ?$

$P(X_2 = 10 \mid G_1 = 1) = 0.0089$ (From part (a) and (b))

Similarly

$P(X_3 = 10) \mid G_1 = 1) = 0.0089$

$P(X_2 = 10 \mid G_1 = 2) = 0.0356$ (From part (b))

Similarly $P(X_3 = 10) G_1 = 2) = 0.0356$

Hence,

$$P(G_1 = 1 \mid X_2 = 10, X_3 = 10) = \frac{(0.0089 * 0.0089)\frac{1}{2}}{(0.0089 * 0.0089)\frac{1}{2} + (0.0356 * 0.0356)\frac{1}{2}}$$

$$= \frac{0.000079}{0.000079 + 0.00126} = \boxed{0.0589}$$

Hence

$P(G_1 = 2 \mid X_2 = 10, X_3 = 10) = 1 - 0.0589$

$= \boxed{0.9411}$

5. Belief Propagation Implementation (25pt)

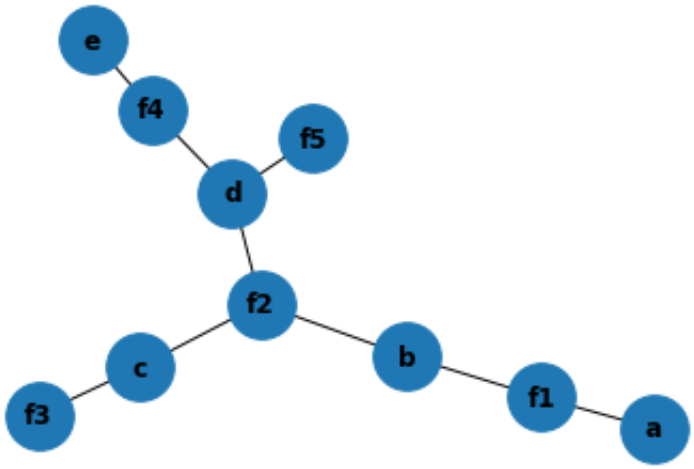
Implement the Sum-Product version of Belief Propagation in Python (using the networkx package to represent the factor graph) to compute the marginal distribution P(A = a,B = b) ∀a,b for the factor graph shown in Figure 3. That is, use the graph object from networkx to define the factor graph and implement the Sum-Product algorithm to pass messages. The support of the corresponding random variables A, B, C, D and E are {1, 2}. The factors are as follows:

1. $f_1(a, b) = a * b$ (for example, $f_1(a = 1, b = 2) = 1 * 2 = 2$)
2. $f_2(b, c, d) = 2 * (5 - b * c) - d + 1$
3. $f_4(d, e) = d * e$
4. $f_3(c) = 3 - c$
5. $f_5(d) = 3 - d$

In [9]:
```
import networkx as nx
import numpy as np
```

In [10]:
```
# Create empty graph object
G = nx.Graph()

# Add edges
edge_list = [('f3','c'),('c','f2'),('e','f4'),('f4','d'),('f5','d'),('
d','f2'),('f2','b'),('b','f1'),('f1','a')]
G.add_edges_from(edge_list)
```

In [11]:
```
import matplotlib.pyplot as plt
plt.subplot(111)
nx.draw(G, with_labels=True, node_size = 1000, font_weight='bold', fon
t_color='k')
plt.show()
```



In [12]:
```
leaf_nodes = []
for node in G.nodes():
    if G.degree(node) == 1:
        leaf_nodes.append(node)
leaf_nodes
```

Out[12]:
```
['f3', 'e', 'f5', 'a']
```

In [13]:
```
root = 'a'
messages = {edge for edge in nx.dfs_tree(G, root).edges}
for leaf in leaf_nodes:
    if leaf != root:
        cand_messages = {edge for edge in nx.dfs_tree(G, leaf).edges}
        messages = messages.union(cand_messages)
messages = {message:None for message in messages}
messages
```

Out[13]:
```
{('f2', 'c'): None,
 ('d', 'f5'): None,
 ('f1', 'a'): None,
 ('d', 'f2'): None,
 ('f3', 'c'): None,
 ('d', 'f4'): None,
 ('f2', 'd'): None,
 ('f5', 'd'): None,
 ('e', 'f4'): None,
 ('b', 'f2'): None,
 ('f4', 'e'): None,
 ('c', 'f2'): None,
 ('f4', 'd'): None,
 ('f1', 'b'): None,
 ('a', 'f1'): None,
 ('b', 'f1'): None,
 ('f2', 'b'): None,
 ('c', 'f3'): None}
```

In [14]:
```
def f1(a, b):
    result = np.zeros(a.shape + b.shape)
    for i, va in enumerate(a):
        for j, vb in enumerate(b):
            result[i,j] = va*vb
    return result

def f2(b,c,d):
    result = np.zeros(b.shape + c.shape + d.shape)
    for i, vb in enumerate(b):
        for j, vc in enumerate(c):
            for k, vd in enumerate(d):
                result[i,j,k] = 2 * (5 - vb * vc) - vd + 1
    return result

def f3(c):
    return 3 - c

def f4(d, e):
    result = np.zeros(a.shape + b.shape)
    for i, vd in enumerate(d):
        for j, ve in enumerate(e):
            result[i,j] = vd*ve
    return result

def f5(d):
    return 3 - d
```

In [65]:
```
a = np.array([1,2])
b = np.array([1,2])
c = np.array([1,2])
d = np.array([1,2])
e = np.array([1,2])
print('f1')
print(f1(a,b))
print('f2')
print(f2(b,c,d))
print('f3')
print(f3(c))
print('f4')
print(f4(d,e))
print('f5')
print(f5(d))
```

```
f1
[[1. 2.]
 [2. 4.]]
f2
[[[8. 7.]
  [6. 5.]]

 [[6. 5.]
  [2. 1.]]]
f3
[2 1]
f4
[[1. 2.]
 [2. 4.]]
f5
[2 1]
```

In [66]:
```
messages[('e', 'f4')] = np.array([1, 1]) # Initial Value
messages[('f4', 'd')] = f4(d, e) @ messages[('e', 'f4')]
messages[('f5', 'd')] = f5(d)
messages[('d', 'f2')] = messages[('f4', 'd')] * messages[('f5', 'd')]
messages[('f3', 'c')] = f3(c)
messages[('c', 'f2')] = messages[('f3', 'c')]
messages[('f2', 'b')] = f2(b, c, d) @ (messages[('c','f2')] * messages
[('d','f2')])
messages[('b', 'f1')] = messages[('f2', 'b')]
messages[('f1', 'a')] = f1(a, b) @ messages[('f2', 'b')]
messages[('a', 'f1')] = np.array([1, 1]) # Initial Value
messages[('f1', 'b')] = f1(a, b) @ messages[('a', 'f1')]
messages[('b', 'f2')] = messages[('f1', 'b')]
messages[('f2', 'c')] = f2(b, c, d) @ (messages[('b', 'f2')] * message
s[('d', 'f2')])
messages[('c', 'f3')] = messages[('f2', 'c')]
messages[('f2', 'd')] = f2(b, c, d) @ (messages[('b', 'f2')] * message
s[('c', 'f2')])
messages[('d', 'f5')] = messages[('f2', 'd')] * messages[('f4', 'd')]
messages[('d', 'f4')] = messages[('f2', 'd')] * messages[('f5', 'd')]
messages[('f4', 'e')] = f4(d, e) @ messages[('d', 'f4')]
for k, v in messages.items():
    print(f'{k}:')
    print(v)
    print()
```

```
('f2', 'c'):
[[396. 288.]
 [288.  72.]]

('d', 'f5'):
[[270. 396.]
 [198. 108.]]

('f1', 'a'):
[[342. 162.]
 [684. 324.]]

('d', 'f2'):
[6. 6.]

('f3', 'c'):
[2 1]

('d', 'f4'):
[[180.  66.]
 [132.  18.]]

('f2', 'd'):
```

```
[[90. 66.]
 [66. 18.]]

('f5', 'd'):
[2 1]

('e', 'f4'):
[1 1]

('b', 'f2'):
[3. 6.]

('f4', 'e'):
[[444. 102.]
 [888. 204.]]

('c', 'f2'):
[2 1]

('f4', 'd'):
[3. 6.]

('f1', 'b'):
[3. 6.]

('a', 'f1'):
[1 1]

('b', 'f1'):
[[138. 102.]
 [102.  30.]]

('f2', 'b'):
[[138. 102.]
 [102.  30.]]

('c', 'f3'):
[[396. 288.]
 [288.  72.]]
```

For joint probability of P(a,b), we next take the product of messages incoming to that section of graph, in this case only m(b, f2) and then normalize by diving by the sum of the resulting product.

In [67]:
```
messages[('b', 'f2')] / np.sum(messages[('b', 'f2')])
```

Out[67]:
```
array([0.33333333, 0.66666667])
```

Here, we see a preference for 2 over 1 in the joint probability.

In [101]:
```
inc_msgs = {}
for node in G.nodes:
    if 'f' in node:
        continue
    inc_msgs[node] = []
    for key, dist in messages.items():
        if key[1] == node:
            inc_msgs[node].append(dist)

marginals = {}
for key, dists in inc_msgs.items():
    dist_prod = 1
    if len(dists) > 1:
        for dist in dists:
            dist_prod = dist * dist_prod
    else:
        dist_prod = dists
    marginals[key] = dist_prod/np.sum(dist_prod)

print('Marginals:')
for k, v in marginals.items():
    print(f'{k}:')
    print(v)
    print()
```

```
Marginals:
c:
[[0.45833333 0.16666667]
 [0.33333333 0.04166667]]

e:
[[[0.27106227 0.06227106]
  [0.54212454 0.12454212]]]

d:
[[0.375 0.275]
 [0.275 0.075]]

b:
[[0.27380952 0.4047619 ]
 [0.20238095 0.11904762]]

a:
[[[0.22619048 0.10714286]
  [0.45238095 0.21428571]]]
```

5.1. Here we used a dictionary keyed with tuples of (origin, destination) to represent the messages. We experiemented with a dictionary of dictionaries, but while likely viable, it proved confusing. Distributions were stored as numpy arrays to allow for matrix math. Note - this algorithm should probably have been coded as a series of recursive sums and products over the graph, gathering incoming messages and multiplying them. We tried constructing a class to track state of such a recursive process, but got hopeless confused, especially with all the changing matrix math. I expect that this was the expected solution. I was thinking of experiementing with numed tuples to for ditributions to help with the math, but I couldn't get it working, and had to move on to other work.

5(b) If outgoing message is connected to f factors, then its complexity has a support of K values from a variable node.

Variable Messages are product of all incoming message except from receiving factor

- f_i assumes $i = 1$ to $f-1$
  (f not considered)

$$\mu_x \to f(x = x) = \pi \qquad \mu_{f_i} \to x \; (X = \mu)$$
$$f \in \{n(x) \mid F\}$$

given: x has k values

$$\mu_x \to f(x = x) \; ---- \; (k \text{ times})$$

Assuming f to be receiving factor

→ Product has $(f-1)$ terms and x has k values

∴ Total complexity $= k(f-1)$

(c) Message from factors sum out all variables except receiving one

(V-1) variables at a time assumes K each variable.

$$\mu_{f \to V_1}(x = v) = \sum_{V_2=1}^{K} \sum_{V_3=1}^{K} \cdots \sum_{V_n=1}^{K} f(V_1, V_2, \cdots V_n) \prod_{V_2}^{V_n - 1} \mu_{V_2 \to f}(V_1)$$

for K additions summed over $V_i = 1$ to k

$\therefore$ $K \cdot K \cdots (V-1)$ gives K^{V-1} additions and $(V-1)$ multiplication for each addition.

6. Metropolis-Hastings Sampling (15pt)

Let $X \sim \pi$ be a distribution. To create a Monte Carlo estimate of the expectation of some function of X, i.e., $E\pi[f(X)]$, we will do Metropolis-Hastings (MH) MCMC sampling. For this, we start with an initial $x = x_0$ and do the following:

- $x' \sim g(\cdot|x)$
- $\alpha = min\left(1, \frac{\pi(x)g(x'|x)}{\pi(x')g(x|x')}\right)$
- With probability α accept x' as the next sample x_t.
- Set $x = x'$ and repeat.

An example of the proposal distribution is $g(x'|x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x'-x)^2}{\sigma^2}}$

1. Implement the MH sampler for estimating $E[(X^2 + 10)x]$ when $\pi(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-3)^2}$

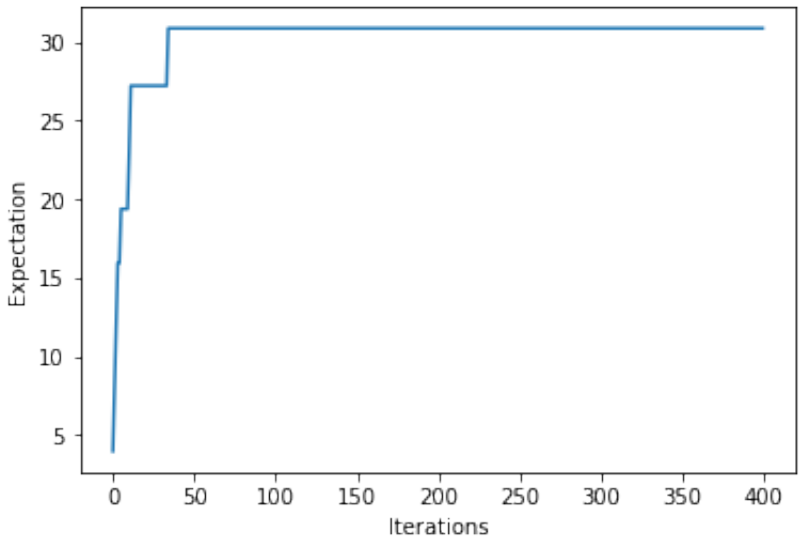
In [733]:
```
from math import sqrt, pi, exp
import numpy as np

def pi_dist(x):
    '''Target distribution to sample from'''
    return 1 / sqrt(2 * pi) * exp(-(x - 3)**2)

def g_dist(x, x_prm, sigma=1):
    return 1 / (sigma * sqrt(2 * pi)) * exp(-(x_prm - x)**2/(sigma**2)
)

def alpha(x, x_prm):
    return min(1, pi_dist(x) / pi_dist(x_prm) * g_dist(x, x_prm) / g_d
ist(x_prm, x))

def f_of_X(x):
    return (pi_dist(x)**2 + 10) * x

def metropolis_hastings(iterations=100):
    x = 0
    x_prm = 0
    samples = np.zeros((iterations, 1))
    expectations = np.zeros((iterations, 1))
    for i in range(iterations):
        x_prm = x + g_dist(x, x_prm)
        a = alpha(x, x_prm)
        if a < np.random.random():
            x = x_prm
        samples[i] = np.array([x])
        expectations[i] = f_of_X(x)
    return(samples, expectations)
```

In [734]:
```
iters = 400
samples, expectations = metropolis_hastings(iterations=iters)
```

1. Plot the estimate as a function of the number of samples used in the estimation.

In [735]:
```
import matplotlib.pyplot as plt

print(f'Final Expectation Estimate: {expectations[-1:]}')
plt.plot(range(iters), expectations)
plt.ylabel('Expectation')
plt.xlabel('Iterations')
plt.show()
```

```
Final Expectation Estimate: [[30.88089218]]
```

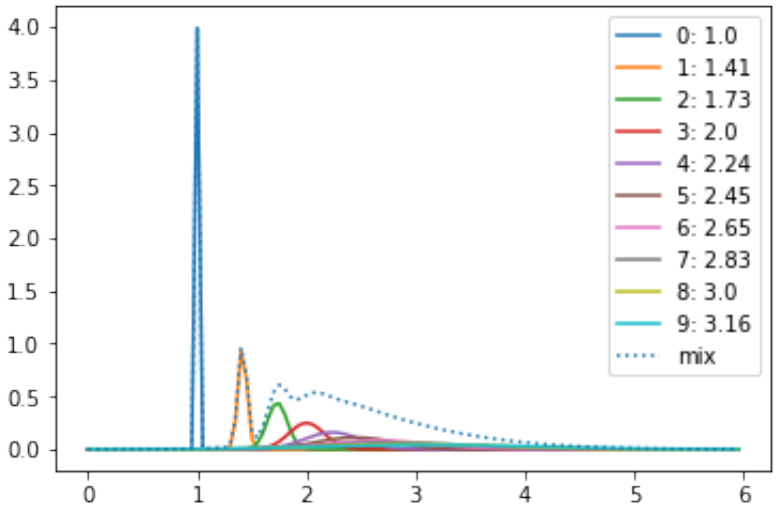


7. Gibbs Sampling (15pt)

Let the distribution $\pi(x) = \sum_{k=1}^{10} \lambda_k \mathcal{N}(x|\mu = \sqrt{k}, \sigma = 0.01k^2)$, with some arbitrarily chosen λ_k's that sum up to 1. Implement the Gibbs sampling procedure to estimate the mean. Report the mean values computed using the first 500 samples and the next 5000 samples.

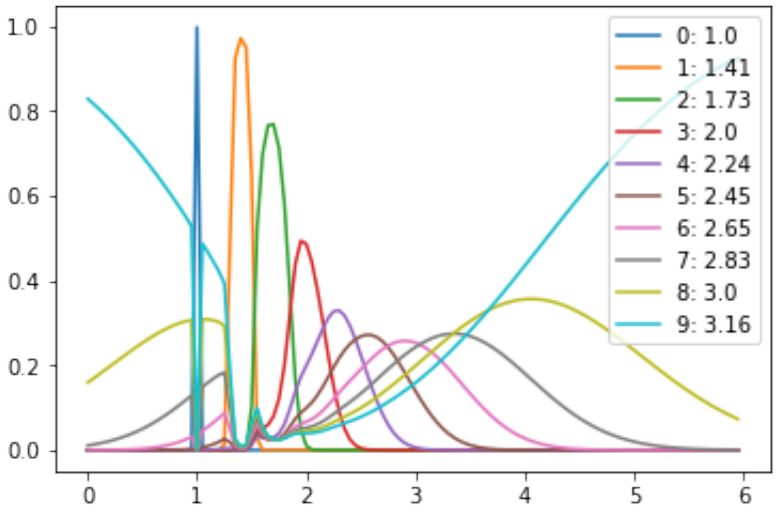
In [1210]:
```
# Visualize univariate mixture of 10 gaussians

import matplotlib.pyplot as plt

lams = [.1,.1,.1,.1,.1,.1,.1,.1,.1,.1]
mus = [sqrt(k) for k in range(1,11)]
sigmas = [.01*k**2 for k in range(1,11)]
xs = np.arange(0,6,.05)

for k, (lam, mu, sigma) in enumerate(zip(lams, mus, sigmas)):
    plt.plot(xs, lam * norm.pdf(xs, mu, sigma), label=f'{k}: {mu:.3}'
)

mix_pdf = np.zeros(len(xs))
for i, x in enumerate(xs):
    mix_pdf[i] = y_given_x(x, lams, mus, sigmas)

plt.plot(xs, mix_pdf, label = 'mix', linestyle = 'dotted')
plt.legend()
```

Out[1210]: <matplotlib.legend.Legend at 0x1a1ddd7f10>



In [1325]:
```
# Visualize latent assignment variables (z) for each gaussian over a
range of x

p_k_given_x = np.zeros([len(xs), len(mus)])
for i in range(len(xs)):
    p_k_given_x[i] = responsibility(xs[i], lams, mus, sigmas)

for k, mu in enumerate(mus):
    plt.plot(xs, p_k_given_x[:,k], label=f'{k}: {mu:.3}')
plt.legend()
```

Out[1325]:
```
<matplotlib.legend.Legend at 0x1a1eb54a10>
```

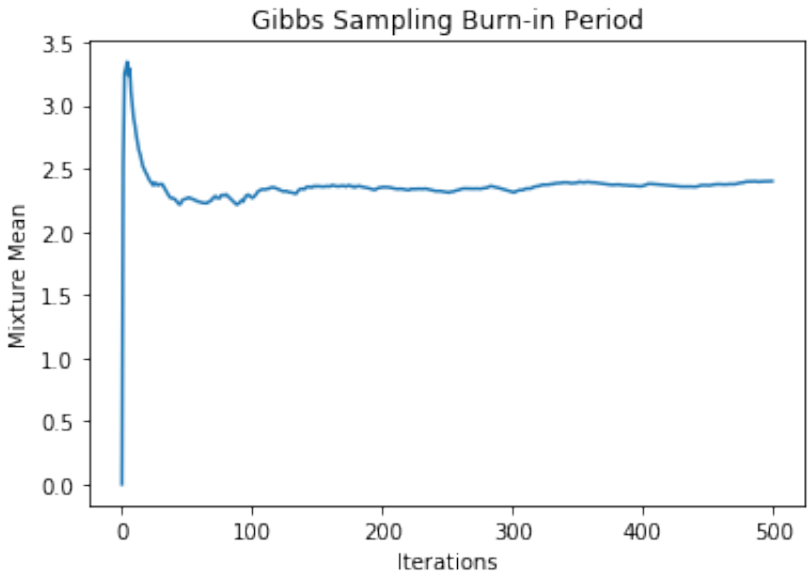


In [1400]:
```
from scipy.stats import norm
from scipy.stats import multinomial

def mix_given_x(x, lams, mus, sigmas):
    result = sum([lam * norm.pdf(x, mu, sigma) for lam, mu, sigma in
zip(lams, mus, sigmas)])
    return result

def pi_sample(lams, mus, sigmas):
    result = sum([lam * norm.rvs(mu, sigma) for lam, mu, sigma in zip
(lams, mus, sigmas)])
    return result

def sample_x_given_z(z, lams, mus, sigmas):
    result = norm.rvs(mus[int(z)], sigmas[int(z)])
    return result

def responsibility(x, lams, mus, sigmas):
    resp = np.zeros(len(mus))
    for k in range(len(mus)):
        resp[k] = lams[k]*norm.pdf(x, mus[k], sigmas[k]) / mix_given_
x(x, lams, mus, sigmas)
    return resp

def sample_z(responsibilities):
    return np.argmax(multinomial.rvs(1, responsibilities))

def mu_est(xs):
     return np.cumsum(xs)/np.arange(1,len(xs)+1)
```

In [1396]:
```
iterations = 5500
xs = np.zeros(iterations)
zs = np.zeros(iterations)
for i in range(1, iterations):
    resps_given_x = responsibility(xs[i-1], lams, mus, sigmas)
    zs[i] = sample_z(resps_given_x)
    xs[i] = sample_x_given_z(zs[i], lams, mus, sigmas)
```

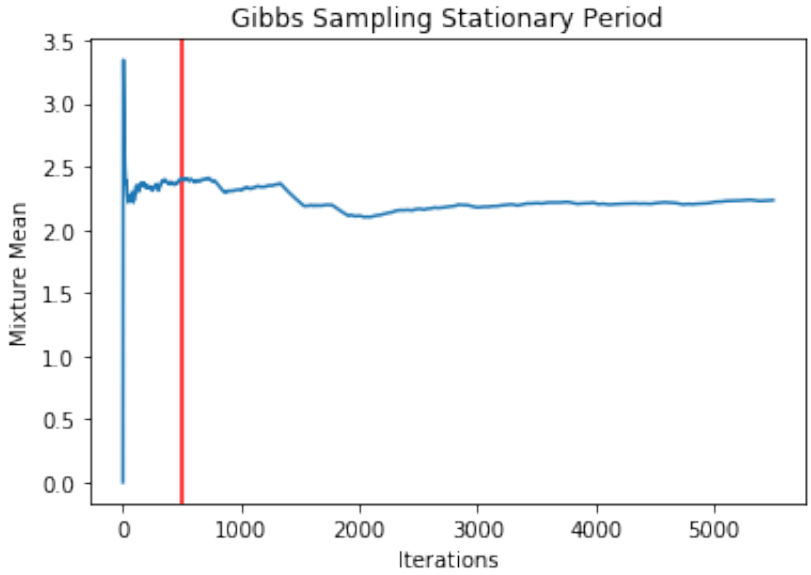
In [1397]:
```
mixture_mu_estimates = mu_est(xs)

plt.title('Gibbs Sampling Burn-in Period')
plt.xlabel('Iterations')
plt.ylabel('Mixture Mean')
plt.plot(range(500), mixture_mu_estimates[:500])
print('Burn-in Period Mean:', np.mean(mixture_mu_estimates[:500]))
```

```
Burn-in Period Mean: 2.3580368923742006
```



In [1399]:
```
plt.title('Gibbs Sampling Stationary Period')
plt.xlabel('Iterations')
plt.ylabel('Mixture Mean')
plt.axvline(500, color='r')
plt.plot(range(iterations), mixture_mu_estimates)
print('Stationary Period Mean:', np.mean(mixture_mu_estimates[500:]))
```

```
Stationary Period Mean: 2.2215600669108566
```



In [ ]:
```

```