# Graphical Models Assignment 1

## Team Name:

- Grapgical_Stars

## Team Members:

- Marika Paraskevopoulou
- Eoghan Flanagan
- Jonathan Smith
- Klaudia Ludwisiak
- Seyd Anane

## Work Breakdown:

We approached the questions in groups of two or three, and then came together as a team to brainstorm ideas and compare answers where needed.

The following table shows in green which people took primary responsibility for which questions. It should be emphasised that for the harder questions the entire team contributed ideas.

| Question | Eoghan | Klaudia | Marika | Jonathan | Seyd |
|---|---|---|---|---|---|
| 2.5 | | | | | |
| 2.6 | | | | | |
| 2.7 | | | | | |
| 2.9 | | | | | |
| 2.10 | | | | | |
| 3.3 | | | | | |
| 3.4 | | | | | |
| 3.8 | | | | | |
| 3.9 | | | | | |
| 3.11 | | | | | |
| 3.12 | | | | | |
| 3.13 | | | | | |
| 3.14 | | | | | |
| 3.15 | | | | | |
| 3.17 | | | | | |
| 3.21 | | | | | |
| 3.22 | | | | | |
| 3.23 | | | | | |
| 3.24 | | | | | |
| 3.25 | | | | | |

Exercise 2.5.

Assume the graph is provided as a list L of pairs (p,c) of (parent, child) pairs in the directed graph. Given a set S of nodes for which we need to calculate the ancestors, find all parents p(S) of S (by iterating through L and matching the elements of S with the children vertices of L). If p(S) is empty we are done. Otherwise repeat the process and compute p(p(S)). Since the graph is acyclic (and presumably finite!) this process will eventually terminate. Then the ancestors of S are p(S) U p(p(S)) U p(p(p(S)))).... In pseudocode -

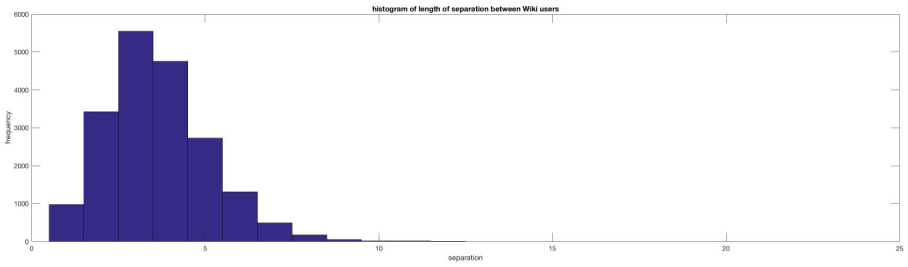
```
Def get_ancestors(T):
        T_parents={}
        For (p,c) in T:
                If c in T:
                        T_parents.append(p)

        If T_parents={}:
                return({})
        Else:
                return(T parents U get_ancestors(T_parents))
```

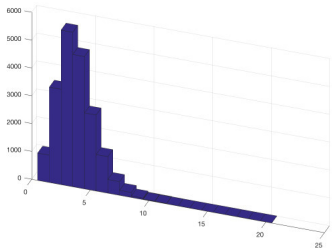
Alternatively, if the graph is provided as an adjacency matrix A(i,j) where A(i,j)=1 if i is the parent of j, 0 otherwise, compute as follows. A^2(i,j) = 1 iff i is the parent of a parent of j by basic properties of matrix multiplication. Therefore the $j^{th}$ column of the matrix reads 1 in the $i^{th}$ row if i is the ancestor of j and zero otherwise. Compute successive powers $A^k$ of A until the number of nonzero entries in the columns of $A^k$ corresponding to S stops increasing. The nonzero rows (in those columns) are then the ancestors of S.

```
%Graphical Models Q2.6
%script loads sparse adjacency matrix of connections between Wikipedia
%authors and outputs the frequency of the shortest path between them

%SETUP
%Sparse double matrix of wiki results
load('WikiAdjSmall.mat');
F=full(A);  %convert sparse matrix to full matrix

s=20;                %initialise separation length (s)
k=linspace(1,s,s)';  %vector of separation length (s)
Path=zeros(1,s)';    %will store frequency of each separation length

[i,j]=find(F); %(returns ith and jth position of nonzero elements of
 A)
Path(1,1)=0.5*length(i);%counts amount of unique entries of shortest
 path length =1.
W=F; %initialises a matrix which will store all the already counted
 'shortest paths'

%CALCULATIONS

for c=2:length(k)
   [ik,jk]= find((F^k(c,1))==1);
   path=0;
   for b=1:length(ik)
       if W(ik(b,1),jk(b,1))==0
        path=path+1;
        W(ik(b,1),jk(b,1))=1;
       end
   end
Path(c,1)=0.5*path;
end
%explanation of the above:
%code counts number of elements in adjacency matrix which are exactly
 equal to 1
%this corresponds to wiki authors for whom the shortest path is of
 length k
%ensures paths of non minimal length are omitted, by checking if
 shorter
%path existed (recorded in matrix W)
%Adjacency matrix is symmetrical, thus summation method above "double
%counts" paths and this is why need to divide by two.


%plot
figure;
subplot(2,1,1);
bar(Path,1);   %could produce ordinary histogram but will also make a
 3D one
xlabel('separation');
ylabel('frequency');
```

```
title('histogram of length of separation between Wiki users');
subplot(2,1,2);
bar3(Path,1);
view(-70,30);

%check
if 0.5*length(find(W))==sum(Path)
disp('correct')
else
    disp('incorrect')
end
```

```
correct
```




*Published with MATLAB® R2016b*

**Ex 2.7**:

Implemented in Python using numpy and pandas the algorithm returns the set of unique maximal cliques (super sets). Using Cliques.mat (e.g. [1,3,5,6,8,10],[7,8]...) the 100 non-maximal cliques on the graph are converted into an 'NxD' binary bit form matrix (**M**). This is based on the position of each of the nodes of the non-maximal clique.

Matrix **M** [100x10] (referred to as 'binmatrix' in the code) represents all of the non-maximal cliques and is shown below e.g. the first row represents the clique [1,3,5,6,8,10].

$$M = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ . & . & . & . & . & . & . & . & . & . \end{bmatrix}$$

**M** is then multiplied by its transpose (**A = MM**$^{\mathsf{T}}$,) where the diagonal elements *reflect the degree of each node. The off diagonal elements contain the number of cliquos that nodes i and j jointly inhabit* [1]. **A** [100x100] is a square and symmetric matrix ('mattranspose').

The algorithm (for loop) iterates once down matrix **A** comparing the diagonal elements to each $ij^{\text{th}}$ elements in the corresponding row. If the diagonal element (degree of node) is equal to an element in the row the relevant non-maximal clique, is deemed redundant (as it is a subset of the diagonal super set clique).

For example, if the diagonal element $a_{2,1}$ is equal to $a_{2,2}$ then it it is eliminated (subset of the corresponding superset clique).

The cliques (still in binary form) are then appended to an new array as **N**.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

**N** is then converted from binary to number representation (using np.arange) yielding the result: **[119, 447, 463, 487, 703, 751, 755, 765, 831, 863, 886, 893, 954, 983, 1006]**

```
1
2  import pandas as pd
3  import numpy as np
4  df = pd.read_csv("cliques.csv")
5  narr = np.array(df)
6  binmatrix = np.nan_to_num(narr)
7  #XXt
8  mattranspose = binmatrix.dot(binmatrix.T)
9  mt = np.array(mattranspose)
10 diag = mattranspose.diagonal()
11 #fill diagnols with 0
12 np.fill_diagonal(mt,0)
13 maximal = []
14 #loop over the xxt array (without diag)
15 for i in range(len(diag)):
16     if diag[i] in mt[i,:]:
17         pass
18     else:
19         maximal.append(binmatrix[i])
20 new = np.array(maximal)
21 number = new.dot(1 << np.arange(new.shape
22 [-1] - 1, -1, -1))
23 print (sorted(number))
```

Listing 1. Python Ex 2.7

[1] http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook

# Exercise 2.9

Label the N/3 subsets of nodes A, B, C, …  Pick one node from each subset, e.g. {A1, B1, C1, …}.

- The nodes {A1, B1, C1, …} are all connected to each other because they are all in different subsets (see question setup).
- There are no other nodes that are connected to all of {A1, B1, C1, …}.  To see this, consider that:
    - A1 is not connected to any other nodes in subset A.
    - B1 is not connected to any other nodes in subset B.
    - and so on…

So {A1, B1, C1, …}, or indeed any similar selection, fulfils the conditions for being a maximal clique.

The number of such maximal cliques is the number of ways of choosing one node from three, from each of N/3 groups, i.e. $3^{\frac{N}{3}}$.

# Exercise 2.10

We can use an undirected graph to represent who was present in the room with whom during the evening. The nodes represent the people and the edges represent a simultaneous presence in the room. So if two people were present simultaneously at any point in time during the evening, an edge is drawn between their respective nodes.

A group of people that are all present at the same time will therefore be represented by a maximal clique, as they are all 'present' with each other. If more people arrive, the maximal clique grows. However, if anybody leaves, a new maximal clique is only formed when a new, different, person arrives.

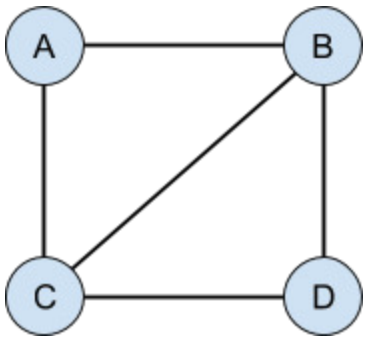
To illustrate this, consider the following simple example with four people, A, B, C and D during the evening. The arrivals and departures are as follows, in time order:

- {A, B, C} arrive
- A leaves
- D arrives
- {B, C, D} leave

The set {A, B, C} overlap in time, and the set {B, C, D} overlap in time.

We therefore have two maximal cliques in the associated undirected graph, one that is formed just before A leaves and one that is formed just before {B, C, D} leave.

The associated undirected graph would look as follows. The two maximal cliques corresponding to the groups that overlap in time can be seen:

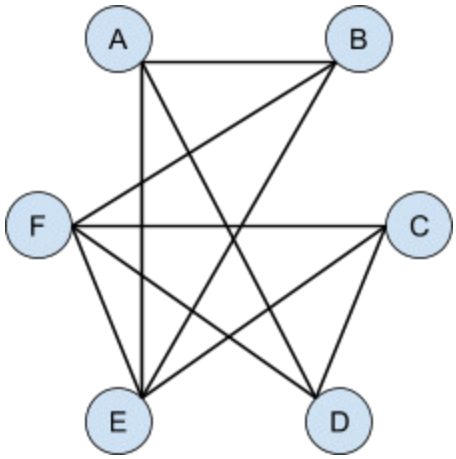



So the approach used to answer this question is:

1. Draw the undirected graph based on the guests' testimonies, including the one false testimony.
2. Find the maximal cliques.

3. Use the constraint that no guest re-entered the room to order the cliques in such a way that it is possible to transition between the cliques only by guests leaving and new (previously unseen) guests arriving.
4. Find which testimony, if relaxed, allows the constraint in (3) to hold.

The graph is shown below, with maximal cliques
{A, B, E}, {B, E, F}, {C, D, F}, {C, E, F}, {A, D}




Putting the maximal cliques into a clique matrix we get the following, with red signifying 1 and green signifying 0. The cliques have been numbered to make the following steps clearer.

| | | Clique number | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Person | A | | | | | |
| | B | | | | | |
| | C | | | | | |
| | D | | | | | |
| | E | | | | | |
| | F | | | | | |

Next, starting with clique 4 in the middle, the other cliques are added one-by-one keeping adjacent cliques as similar as possible. The choice of clique 4 is arbitrary, but members C and F were present according to two different statements. This tries to meet the condition of guests not leaving and returning.

This reordered representation could be seen as one possible time-order of the evening, e.g. from left to right.  Note that the groups of red squares generally now form single contiguous horizontal blocks, showing how individual guests do not return after they have left.

| | | Clique number | | | | |
|---|---|---|---|---|---|---|
| | | 5 | 3 | 4 | 2 | 1 |
| Person | A | | | | | |
| | B | | | | | |
| | C | | | | | |
| | D | | | | | |
| | E | | | | | |
| | F | | | | | |

Note that in this clique order, person A is present at two different times, which violates the condition in the question.  To reconcile this, either cell (A,5) or cell (A,1) must be changed.

Cell (A,5) can be changed by modifying the statement in the question.

> *4. D: I was present in the room with A and F*

To:

> *4. D: I was present in the room with ~~A and~~ F*

This corrects this issue and now meets the constraint in the question.

Cell (A,1) can only be changed by modifying two statements (1 and 2).

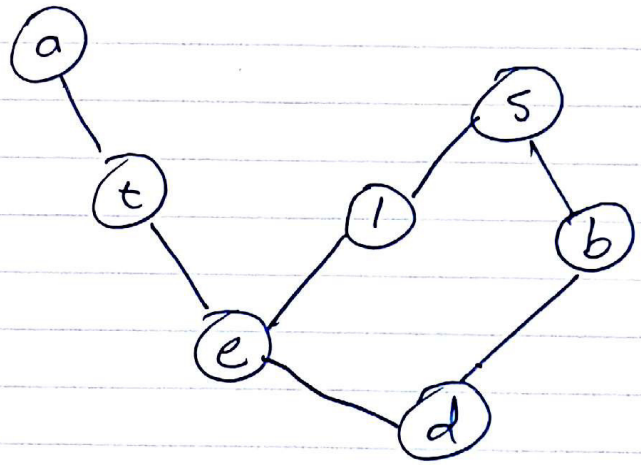
**Answer: The false testimony is number 4.**

Exercise 3.3 - 1

1. Tuberculosis $\perp\!\!\!\perp$ smoking | shortness breath

(in graph terms) $t \perp\!\!\!\perp s \mid d$

d is a collider => conditioning on a collider induces dependency between its parents => we have the following undirected graph structure




=> <u>False</u>

2. Lung cancer $\perp\!\!\!\perp$ bronchitis | smoking

(in graph terms) $l \perp\!\!\!\perp b \mid s$

Conditioning on a non-collider, s, breaks the link between l, b. Since d is a collider it blocks the link between

Exercise 3.3 - 2

1 and b. => True

3. visit to Asia ⫫ smoking | lung cancer

Conditioning on l, a non-collider breaks the link between s and a since the causal link cannot pass through the collider d

=> True

4. visit to Asia ⫫ smoking | lung, dyspnea

d is a collider => conditioning on it induces dependency between its parents

=> False

3.4

$P(t=tr) = p(t=tr \mid a=tr)\, p(a=tr) + P(t=tr \mid a=fa)\, P(a=fa) =$

$= 0.0104$

$P(l=+r) = p(l=+r \mid s=+r)\, p(s=+r) + p(l=tr \mid s=fa)\, p(s=fa) =$

$= 0.055$

$P(e=+r) = p(t=tr, l=tr) = 0.0648$

$P(b=tr) = p(b=tr \mid s=+r)\, P(s=tr) + P(b=tr \mid s=fa)\, p(s=fa) = 0.45$

$P(d) = \sum_{e,d} p(d=true \mid e,d) = p(d=true \mid e=tr, d=+r) + p(d=+r \mid e=fa, d=fa)$
$P(d=+r \mid e=tr, d=fa) + p(d=+r \mid e=tr, d=+r)$
$= 0.43$

hence $\underline{P(d) = 0.43}$

Also $p(b=tr \mid s=+r) = 0.6$ and $P(l=tr \mid s=+r) = 0.1$

$P(d=tr \mid e=tr, b=tr) = 0.9$ , $P(d=tr \mid e=tr, b=fa) = 0.7$

and $p(d=tr \mid e=fa, b=tr) = 0.8$ , $P(d=tr \mid e=fa, b=fa) = 0.1$

also $p(e=tr \mid s=true) = 1 - (0.9 \times 0.9896) = 0.1094$

thus $p(d \mid s) = 0.9 * 0.1094 * 0.6 +$
$0.8 * (1-0.1094) * 0.6 +$
$0.7 * (0.1094) * 0.4 +$
$0.1 * (1-0.1094) * 0.1 = 0.5261$

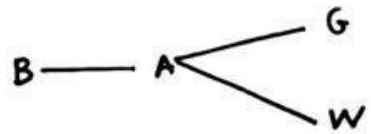
hence $\underline{P(d=true \mid s=true) = 0.5261}$

finally for calculating

$p(d=true \mid s=false)$ we have $P(b \mid s=fa) = 0.3$
$P(l \mid s=fa) = 0.01$
$P(t) = 0.0104$

also $p(e=true \mid s=false) = 1 - (1-0.01) * (1-0.0104) = 0.0203$

hence $P(d \mid s=fa) = 0.9 * 0.0203 * 0.3 + 0.8 * (1-0.0203) * 0.3 +$
$+ 0.7 * 0.0203 * 0.7 + 1 * (1-0.0203) * 0.7 =$
$= 0.3191$, thus $\underline{P(d=true \mid s=fa \mid se) = 0.3191}$




$P(B,A,G,W) = P(A|B)\,P(B)\,P(G|A)\,P(W|A)$.

1 (a) $P(B=tr|W=tr) = \dfrac{P(B=tr, W=tr)}{P(W=tr)} =$

$$= \frac{\sum_{A,G} P(A|B=tr)\,P(B=tr)\,P(G|A)\,P(W=tr|A)}{\sum_{B,A,G} P(A|B)\,P(B)\,P(G|A)\,P(W=tr|A)}$$

$$= 0.0171$$

(b) $P(B=tr|W=tr, G=fa) = \dfrac{P(B=tr, W=tr, G=fa)}{P(W=tr, G=fa)} =$

$$= \frac{\sum_{A} P(A|B=tr)\,P(B=tr)\,P(W=tr|A)\,P(G=fa|A)}{\sum_{B,A} P(A|B)\,P(B)\,P(W=tr|A)\,P(G=fa|A)}$$

$$= 0.0069$$

2 Jefrey's Rule

(a) $P(W|\tilde{W}) = \begin{cases} 0.3, & W=tr \\ 0.7, & W=fa \end{cases}$ and $P(G|\tilde{G}) = \begin{cases} 0.1, & G=tr \\ 0.9, & G=fa \end{cases}$

hence $P(B=tr|\tilde{W}) = \sum_{W} P(B=tr|W)\,P(W|\tilde{W}) =$

$$= P(B=tr|W=tr)\,P(W=tr|\tilde{W}) + P(B=tr|W=fa)\,P(W=fa|\tilde{W}) =$$

$$= 0.0171 * 0.3 + 0.00218 * 0.7 = 0.0021 + 0.0015$$

$$= 0.0067$$

(b) $P(B=tr \mid \tilde{W}, \tilde{G}) = \sum_{W,G} P(B=tr \mid W,G)\, P(W \mid \tilde{W})\, P(G \mid \tilde{G}) =$

$= P(B=tr \mid W=tr, G=tr)\, P(W=tr \mid \tilde{W})\, P(G=tr \mid \tilde{G}) +$

$P(B=tr \mid W=tr, G=fa)\, P(W=tr \mid \tilde{W})\, P(G=fa \mid \tilde{G}) +$

$P(B=tr \mid W=fa, G=fa)\, P(W=fa \mid \tilde{W})\, P(G=fa \mid \tilde{G}) +$

$P(B=tr \mid W=fa, G=tr)\, P(W=fa \mid \tilde{W})\, P(G=tr \mid \tilde{G})$

$= 0.0475 * 0.3 * 0.1 + 0.0089 * 0.3 * 0.9 +$

$0.00089 * 0.7 * 0.9 + 0.0071 * 0.7 * 0.1 =$

$= 0.0014 + 0.0019 + 0.00056 + 0.00049 = 0.0043$

```
%3.8
clear all
import brml.*
[A B G W]=assign(1:4);
yes=2;
no=1;
variable(A).name='alarm';
variable(A).domain={'yes','no'};
variable(B).name='burglar';
variable(B).domain={'yes,no'};
variable(W).name='watson';
variable(W).domain={'yes,no'};
variable(G).name='gibbon';
variable(G).domain={'yes,no'};
pot(B)=array;
pot(B).variables=B
pot(B).table(yes)=0.01;
pot(B).table(no)= 1-pot(B).table(yes);
pot(A)=array;
pot(A).variables=[A B]
pot(A).table(yes,yes)=0.99;
pot(A).table(yes,no)=0.05;
pot(A).table(no,yes)=1-pot(A).table(yes,yes);
pot(A).table(no,no)=1-pot(A).table(yes,no);
pot(W)=array;
pot(W).variables=[W A]
pot(W).table(yes,yes)=0.9;
pot(W).table(yes,no)=0.5;
pot(W).table(no,yes)=1-pot(W).table(yes,yes);
pot(W).table(no,no)=1-pot(W).table(yes,no);
pot(G)=array;
pot(G).variables=[G A]
pot(G).table(yes,yes)=0.7;
pot(G).table(yes,no)=0.2;
pot(G).table(no,yes)=1-pot(G).table(yes,yes);
pot(G).table(no,no)=1-pot(G).table(yes,no);
joinpot=multpots(pot([A B G W]))
pot_BW = setpot(joinpot, W, yes);
prob_BW = condpot(pot_BW, B);
disptable(prob_BW)
pot_BWG = setpot(joinpot, [W G], [yes no]);
prob_BWG = condpot(pot_BWG, B);
disptable(prob_BWG)
pot_BWf = setpot(joinpot, W, no);
prob_BWf = condpot(pot_BWf, B);
disptable(prob_BWf)
pot_BWGy = setpot(joinpot, [W G], [yes yes]);
prob_BWGy = condpot(pot_BWGy, B);
disptable(prob_BWGy)
pot_BWGyn = setpot(joinpot, [W G], [yes no]);
prob_BWGyn = condpot(pot_BWGyn, B);
disptable(prob_BWGyn)
```

```
pot_BWGn = setpot(joinpot, [W G], [no no]);
prob_BWGn = condpot(pot_BWGn, B);
disptable(prob_BWGn)
pot_BWGny = setpot(joinpot, [W G], [no yes]);
prob_BWGny = condpot(pot_BWGny, B);
disptable(prob_BWGny)


pot =

  1×2 array array with properties:

    variables
    table


pot =

  1×2 array array with properties:

    variables
    table


pot =

  1×4 array array with properties:

    variables
    table


pot =

  1×4 array array with properties:

    variables
    table


joinpot =

  2×2×2×2 array array with properties:

    variables: [1 2 3 4]
        table: [2×2×2×2 double]

9.828929e-01
1.710707e-02

ans =

     []
```

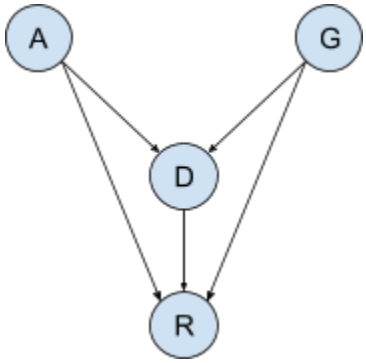
```
9.930840e-01
6.916014e-03

ans =

     []

9.978162e-01
2.183773e-03

ans =

     []

9.524878e-01
4.751221e-02

ans =

     []

9.930840e-01
6.916014e-03

ans =

     []

9.991085e-01
8.914825e-04

ans =

     []

9.928425e-01
7.157547e-03

ans =

     []
```

*Published with MATLAB® R2016b*

# Exercise 3.9

## Part 1

The belief net for the situation is shown below:




This form is arrived at by considering a 'cascade form' belief net of $P(A,G,D,R)$ in that order, and then using the $A \perp G$ condition given in the question to write $P(G \mid A) = P(G)$.

## Part 2

Assuming the distributional information is available, the joint probability distribution $P(A,G,D,R)$ can be found by multiplying together the conditional distributions in the belief net. I.e.:

$$P(A,G,D,R) = P(A)P(G)P(D \mid A,G)P(R \mid A,G,D)$$

Once the joint probability distribution has been calculated, marginalisation can be used to calculate $P(D,R)$ and $P(D)$.

Then the definition of conditional probability can be used to calculate $P(R \mid D)$, i.e..

$$P(R \mid D = drug) = \frac{P(R, D = drug)}{P(D = drug)}$$
$$P(R \mid D = no\ drug) = \frac{P(R, D = no\ drug)}{P(D = no\ drug)}$$

## Part 3

In the same way, though marginalisation of the joint probability distribution $P(R, D = drug, A = young)$ and $P(D = drug, A = young)$ can be calculated.

Then, $P(R \mid D = drug,\ A = young) = \frac{P(R, D = drug, A=young)}{P(D = drug, A=young)}$

# Exercise 3.11

The original model in *demoBurglar.m* makes an assumption in the model setup that 'burglar' and 'earthquake' are independent events.

More formally, considering the joint probability $P(earthquake, burglar, alarm, radio)$ in cascade form in that order, $P(earthquake \mid burglar) = P(earthquake)$ allowed the edge connecting earthquake and burglar to be removed.

In the model setup this information is contained in lines 12-14 of *demoBurglar.m:*

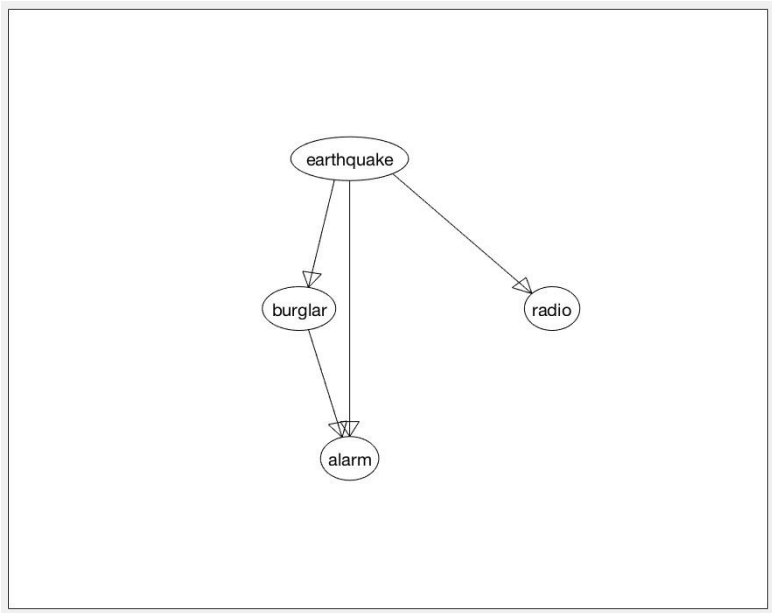
```
pot(burglar).variables=burglar;
pot(burglar).table(yes)=0.01;
pot(burglar).table(no)=0.99;
```

I.e. burglar is not conditionally dependent on any other variables.

Modifying the model as follows (with made-up numbers):

```
pot(burglar).variables=[burglar earthquake];
pot(burglar).table(yes,yes)=0.05;
pot(burglar).table(no,yes)=0.95;
pot(burglar).table(yes,no)=0.01;
pot(burglar).table(no,no)=0.99;
```

Reintroduces the dependency and gives the new belief net, as expected:



Question 3.12

*The code and annotations below provide the solution approach.*

*The code has been tested using example matrices from question 13.13, which are markov equivalent, as well as other matrix pairs where markov equivalence is not preserved. For examples refer to the .m file (MarkovEqTests.m) where test data can be run.*

```
function [sol]=MarkovEq(A,B)
%function checks if two matrices A and B are Markov Equivalent (share same
%skeleton and the same immoralities)
%input:
%A and B should be square matrices of the same size.
%A and B should be constructed with same vertex labels.
%output:
%default sets solution to 1 unless below code sets it to 0,
%effectively checking if Markov equivalence is violated

%check if A and B have same dimensions
if size(A)~=size(B) & size(A,1)==size(A,2)
   disp('A and B should have same dimensions and be square');
   sol=0;
   return;
end

sol=1; %default set to 1 unless below code proves otherwise

%matrices are Markov Equivalent if share skeleton (undirected) and also
%same immoral parents

%1st check skeleton
if issymmetric(A==B)~=1
  disp('not same skeleton so not markov equivalent');
  sol=0;
end

%could have alternative approach for skeleton check where need to check if
%(A+A')==(B+B') this should be all ones if same skeleton

%2nd check for immoralities

m=size(A,2); %each column of matrix

for c=1:m
```

```
 temp=find(A(:,c))'; %find location of all 1s. Transpose for computability.
 if length(temp)>1 %only consider if>1 as then can have immorality on node.
       for i=temp
       for j=temp
          if A(i,c)~=B(j,c) %check if immoralities preserved between A & B
             disp('not same immoralities so not markov equivalent');
             sol=0;
             return;
          end
       end
       end
  end
end
```

```
%Question 3.12
%The code and annotations below provide the solution approach.
%The code has been tested using example matrices from question 13.13,
 which
%are markov equivalent, as well as other matrix pairs where markov
%equivalence is not preserved. For examples refer to the bottom of
 script.


function [sol]=MarkovEq(A,B)
%function checks if two matrices A and B are Markov Equivalent (share
 same
%skeleton and the same immoralities)
%input:
%A and B should be square matrices of the same size.
%A and B should be constructed with same vertex labels.
%output:
%default sets solution to 1 unless below code sets it to 0,
%effectively checking if Markov equivalence is violated


%check if A and B have same dimensions
if size(A)~=size(B) && size(A,1)==size(A,2)
    disp('A and B should have same dimensions and be square');
    sol=0;
    return;
end


sol=1; %default set to 1 unless below code proves otherwise


%matrices are Markov Equivalent if share skeleton (undirected) and
 also
%same immoral parents


%1st check skeleton
if issymmetric(A==B)~=1
  disp('not same skeleton so not markov equivalent');
  sol=0;
end


%could have alternative approach for skeleton check where need to
 check if
%(A+A')==(B+B') this should be all ones if same skeleton


%2nd check for immoralities
```

```
m=size(A,2); %each column of matrix


for c=1:m
 temp=find(A(:,c))'; %find location of all 1s. Transpose for
 computability.
 if length(temp)>1 %only consider if>1 as then can have immorality on
 node.
       for i=temp
       for j=temp
           if A(i,c)~=B(j,c) %check if immoralities preserved between
 A & B
               disp('not same immoralities so not markov equivalent');
               sol=0;
               return;
           end
       end
       end
  end
end

%Markov Equivalent TESTS - run to verify that MarkovEq works

%test 1
simpleA=[0 1 0; 0 0 0; 0 1 0];
simpleB=simpleA;
simpleB2=simpleA;
simpleB2(1,3)=1;
MarkovEq(simpleA,simpleB2)

%test 2 -from notes- known NOT to be MK Equiv
notesA=[0 1 0 0 0; 0 0 1 1 0; 0 0 0 1 0; 0 0 0 0 0;0 0 0 1 0];
notesB=[0 0 0 0 0; 1 0 0 0 0;0 1 0 1 0;0 1 0 0 0;0 0 0 1 0];
MarkovEq(notesA,notesB)

%test 3
load('ABmatrices.mat'); %(need data set in same directory)
MarkovEq(a,b)
```

```
Not enough input arguments.

Error in MarkovEq (line 20)
if size(A)~=size(B) && size(A,1)==size(A,2)
```

*Published with MATLAB® R2016b*

3.13

A.)

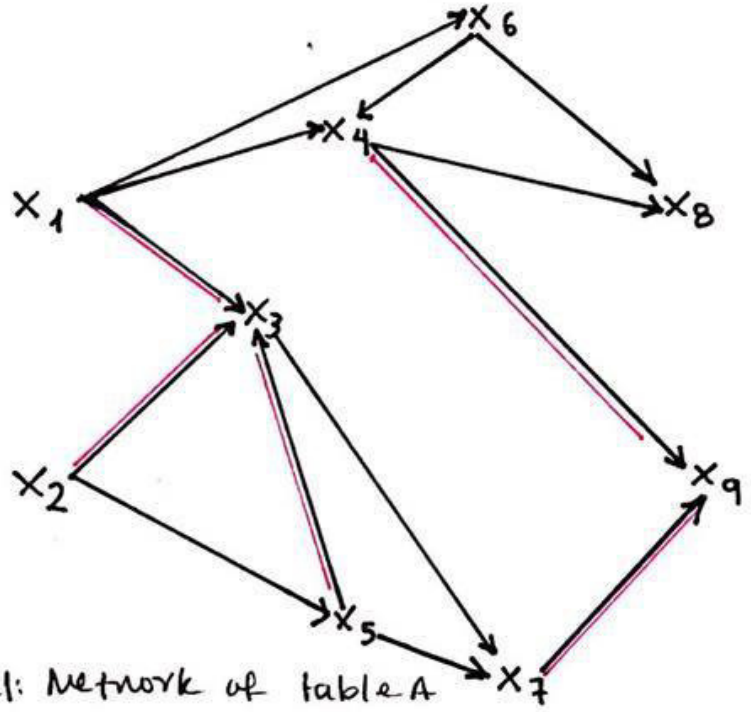



figure1: Network of table A

Immordities

$X_1, X_2 \rightarrow X_3$

$X_1, X_5 \rightarrow X_3$

$X_4, X_7 \rightarrow X_9$

B.)

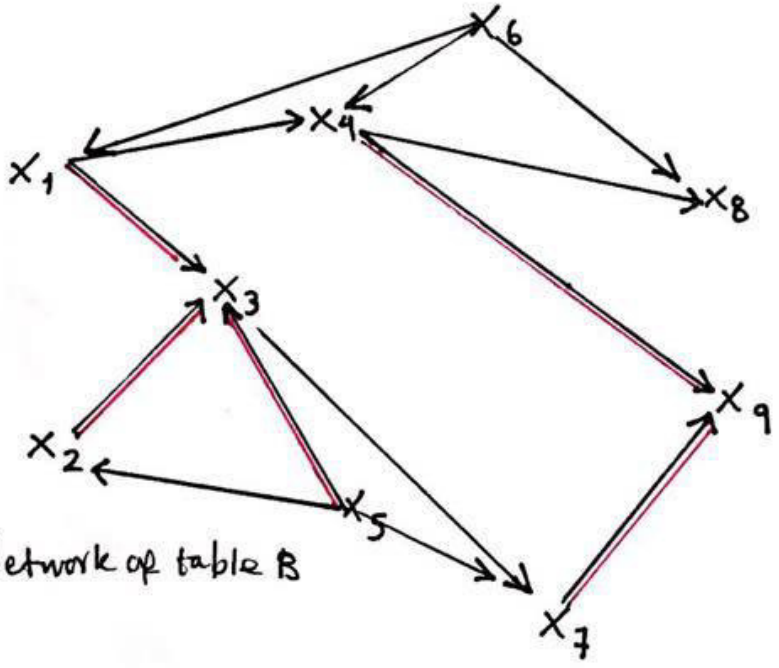



figure 2: Network of table B

Immorallities

$X_1, X_2 \rightarrow X_3$

$X_1, X_5 \rightarrow X_3$

$X_4, X_7 \rightarrow X_9$

Condition one : they have the same skeleton
(we can see it be removing the edges in figure 1+2)

Condition two : they share the same immordities

$\Rightarrow$ Markov equivalent

Finally we use our code from question 3.12 and we conclude also that table A and B are Markov Equivalent.

# Exercise 3.14

Let $c_{ij}$ be the probability that a message can go from $i$ to $j$ in one step

$c_{1,2}(2)=1 \Rightarrow c_{1,2}=1$ or ($c_{1,3}=1$ and $c_{3,2}=1$)

$c_{2,3}(2)=0 \Rightarrow c_{2,3}=0$ and ($c_{2,1}=0$ or $c_{3,3}=0$

Let $\mathcal{E}$ denote the combined evidence that $c_{1,2}(2)$ and $c_{2,3}(2)=0$

$\mathcal{E} \Rightarrow c_{1,2}=1$ and $c_{2,3}=0$ and $c_{2,1}=0$ or

$c_{1,2}=1$ and $c_{2,3}=0$ and $c_{1,3}=0$ or

$c_{1,3}=1$ and $c_{3,2}=1$ and $c_{2,3}=0$ and $c_{3,1}=0$

or $c_{1,3}=1$ and $c_{3,2}=1$ and $c_{1,3}=0$

Note that the fourth option is impossible

Therefore the prior probability $p(\mathcal{E})$ of $\mathcal{E}$ is

$$2\times 0.1\times(0.9)^2+(0.1)^2\times .9$$

$$= 2\times .081 + .009$$

$$= .171$$

Exercise 3.14

$$p(C_{1,2}=1 \mid E) = \frac{p(E \mid C_{1,2}=1)\, p(C_{1,2}=1)}{p(E)}$$

$$= \frac{0.1 \times p(C_{2,3}=0 \text{ and } (C_{2,1}=0 \text{ or } C_{1,3}=0))}{.171}$$

$$= \frac{0.1 \times 0.9 \times (1-.01)}{.171} = \frac{0.1 \times 0.9 \times .99}{.171}$$

$$= 0.521$$

$$p(C_{1,3}=1 \mid E) = \frac{0.1 \times p(C_{3,2}=1 \text{ and } C_{2,3}=0 \text{ and } C_{2,1}=0}{.171}$$

$$= \frac{0.1 \times (0.1 \times 0.9 \times 0.9)}{.171} = .047$$

$$p(C_{23}=1 \mid E) = 0$$

since $C_{2,3}=1$ is inconsistent with the evidence $C_{2,3}(2)=0$

$$p(C_{3,2}=1 \mid E) = \frac{0.1 \times p(C_{1,3}=1, C_{2,3}=0, C_{2,1}=0)}{.171}$$

$$= \frac{0.1 \times 0.1 \times (0.9)^2}{.171} = .047$$

$$p(c_{2,1} = 1 \mid \mathcal{E}) = \frac{0.1 \times 0.1 \times 0.9 \times 0.9}{.171}$$

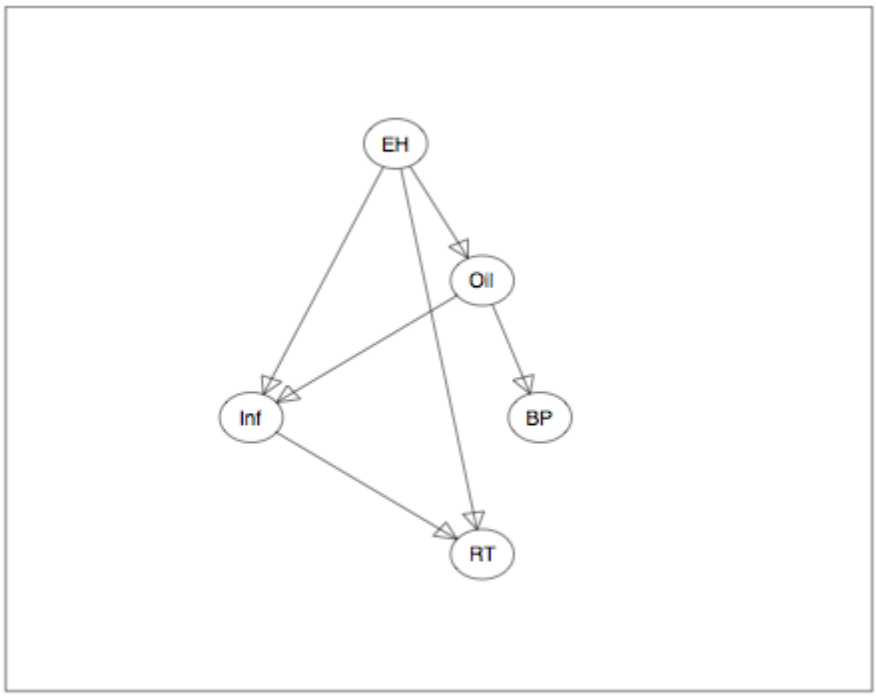
$$= .047$$

$p(c_{3,1} = 1 \mid \mathcal{E}) = 0.1$, the prior, since $c_{3,1} = 1$ and $c_{3,2} = 0$ are both consistent with the evidence

# Exercise 3.15

## Part 1

Using the relationships given in the conditional probability distribution, the following DAG is produced:




## Part 2

Using the BRML toolbox,

$$P(inf = high \mid bp = normal,\ rt = high) = 0.984$$

The code to arrive at this conclusion follows.

```
clear all
import brml.*

% define variables
[oil inf eh bp rt] = assign(1:5);

% define states
% this order is used to avoid the states of bp clashing with the
 states
% of the others.
low = 1;
high = 2;
normal = 3;

% define variable names and domain
variable(oil).name='Oil';
variable(oil).domain = {'low','high'};
variable(inf).name='Inf';
variable(inf).domain = {'low','high'};
variable(eh).name='EH';
variable(eh).domain = {'low','high'};
variable(bp).name='BP';
variable(bp).domain = {'low','high', 'Normal'};
variable(rt).name='RT';
variable(rt).domain = {'low','high'};

% Define and fill potentials
pot(eh) = array;
pot(eh).variables = eh;
pot(eh).table(low) = 0.2;
pot(eh).table(high) = 1 - pot(eh).table(low);

pot(oil) = array;
pot(oil).variables=[oil eh];
pot(oil).table(low,low) = 0.9;
pot(oil).table(low,high)= 0.05;
pot(oil).table(high,:) = 1 - pot(oil).table(low,:);

pot(inf) = array;
pot(inf).variables = [inf oil eh];
pot(inf).table(low,low,low) = 0.9;
pot(inf).table(low,low,high) = 0.1;
pot(inf).table(low,high,low) = 0.1;
pot(inf).table(low,high,high) = 0.01;
pot(inf).table(high,:,:) = 1 - pot(inf).table(low,:,:);

pot(rt) = array;
pot(rt).variables = [rt inf eh];
pot(rt).table(low,low,low) = 0.9;
pot(rt).table(low,low,high) = 0.1;
pot(rt).table(low,high,low) = 0.1;
pot(rt).table(low,high,high) = 0.01;
```

```
pot(rt).table(high,:,:) = 1 - pot(rt).table(low,:,:);

pot(bp) = array;
pot(bp).variables = [bp oil];
pot(bp).table(low,low) = 0.9;
pot(bp).table(low,high) = 0.1;
pot(bp).table(normal,low) = 0.1;
pot(bp).table(normal,high) = 0.4;
pot(bp).table(high,:) = 1 - (pot(bp).table(low,:) +
 pot(bp).table(normal,:));


% construct joint probability distribution
jointpot = multpots(pot([oil inf eh bp rt]));


% do some sanity checks on the joint distribution
% Check marginal distribution of hamburger
epsilon = 1e-15;

% Test 1 - from question "p(eh=low)=0.2"
eh_marginal = sumpot(jointpot, eh, 0);  % sum over everything except
 eh
assert(eh_marginal.table(low) - 0.2 < epsilon, 'Test 1 - Failed');

% Test 2 - from question "p(inf=low|oil=high,eh=high)=0.01"
test_2 = condpot(setpot(jointpot, [oil eh], [high high]), inf);
assert(test_2.table(low) - 0.01 < epsilon, 'Test 2 - Failed');


% draw the belief net (removed for publishing code)
%disptable(jointpot,variable);
%drawNet(dag(pot),variable);


% Now calculate the answer to part 2
% set bp high and rt high
pot_2 = setpot(jointpot, [bp rt], [normal high]);

% We only want the (normalised) P(inf = high) part of this
prob_inf = condpot(pot_2, inf);

disp('p(inf | bp = high, rt = high):')
disptable(prob_inf,variable);
```

```
p(inf | bp = high, rt = high):
Inf  =low   1.525395e-02
Inf  =high  9.847461e-01
```

*Published with MATLAB® R2016b*

# Exercise 3.17

## Part 1:

Substituting the values in from *(3.7.6)* into *(3.7.5)* and marginalising over *b* give:

$$P(a=1,\ c=1) = \left(\tfrac{1}{3}\times\tfrac{1}{4}\times\tfrac{3}{5}\right)+\left(\tfrac{1}{2}\times\tfrac{1}{12}\times\tfrac{3}{5}\right)+\left(\tfrac{15}{40}\times\tfrac{2}{3}\times\tfrac{3}{5}\right) = \tfrac{9}{40}$$
$$P(a=1,\ c=2) = \left(\tfrac{2}{3}\times\tfrac{1}{4}\times\tfrac{3}{5}\right)+\left(\tfrac{1}{2}\times\tfrac{1}{12}\times\tfrac{3}{5}\right)+\left(\tfrac{5}{8}\times\tfrac{2}{3}\times\tfrac{3}{5}\right) = \tfrac{15}{40}$$
$$P(a=2,\ c=1) = \left(\tfrac{1}{3}\times\tfrac{15}{40}\times\tfrac{2}{5}\right)+\left(\tfrac{1}{2}\times\tfrac{1}{8}\times\tfrac{2}{5}\right)+\left(\tfrac{15}{40}\times\tfrac{1}{2}\times\tfrac{2}{5}\right) = \tfrac{6}{40}$$
$$P(a=2,\ c=2) = \left(\tfrac{2}{3}\times\tfrac{15}{40}\times\tfrac{2}{5}\right)+\left(\tfrac{1}{2}\times\tfrac{1}{8}\times\tfrac{2}{5}\right)+\left(\tfrac{5}{8}\times\tfrac{1}{2}\times\tfrac{2}{5}\right) = \tfrac{10}{40}$$

Marginalising $P(a,\ c)$ over c gives:

$$P(a=1) = \tfrac{9}{40}+\tfrac{15}{40} = \tfrac{24}{40}$$
$$P(a=2) = \tfrac{6}{40}+\tfrac{10}{40} = \tfrac{16}{40}$$

Marginalising $P(a,\ c)$ over a gives:

$$P(c=1) = \tfrac{9}{40}+\tfrac{6}{40} = \tfrac{15}{40}$$
$$P(c=2) = \tfrac{15}{40}+\tfrac{10}{40} = \tfrac{25}{40}$$

So, $P(a,\ c) = P(a)P(c)$ for the entire range of a and c, which satisfies the condition for a and c being independent.

# Exercise 3.17

## Part 2

We are given $p(a,b,c) = \frac{1}{Z}\phi(a,b)\psi(b,c)$

Marginalise over b to get:

$$\sum_{p \in dom(b)} p(a, b=p, c) = \frac{1}{Z} \sum_{p \in dom(b)} \phi(a, b=p)\psi(b=p, c)$$

Changing to matrix M and N notation gives

$$p(a=i, c=k) = \frac{1}{Z} \sum_{p \in dom(b)} M_{ip} N_{kp}$$

$$= \frac{1}{Z} \sum_{p \in dom(b)} M_{ip} N^{T}_{pk}$$

This is in the form of a matrix multiplication,

ie $$\frac{1}{Z}\left[MN^{T}\right]_{ik}$$

## Part 3

We are given $MN^T = \vec{m}_o \vec{n}_o^T$

Using the result from part 2,

$$p(a=i, c=k) = \frac{1}{Z}\left[\vec{m}_o \vec{n}_o^T\right] = \frac{m_{o,i}\, n_{o,k}}{Z}$$

where $m_{o,i}$ is the $i^{th}$ element of vector $\vec{m}_o$.
and $n_{o,k}$ is the $k^{th}$ element of vector $\vec{n}_o$.

This is of the form $p(a, c) = f(a)\, g(c)$,
so $a$ and $c$ are independent.

This can also be shown by marginalising over $c$ to get $P(a)$ and over $a$ to get $P(c)$.

Marginalise over $a$ to get:

$$P(c=k) = \frac{1}{Z}\sum_{i \in dom(a)} m_{o,i}\, n_{o,k} = \frac{n_{o,k}}{Z}\sum_{i \in dom(a)} m_{o,i}$$

Marginalise over $c$ to get:

$$P(a=i) = \frac{1}{Z}\sum_{k \in dom(c)} m_{o,i}\, n_{o,k} = \frac{m_{o,i}}{Z}\sum_{k \in dom(c)} n_{o,k}$$

Putting these together gives:

$$P(a=i)\,P(c=k) = \left(\frac{n_{0,k}}{Z}\sum_{i\in dom(a)} m_{0,i}\right)\left(\frac{m_{0,i}}{Z}\sum_{k\in dom(c)} n_{0,k}\right)$$

$$= \frac{m_{0,i}\,n_{0,k}}{Z} \cdot \left[\frac{1}{Z}\sum_{i\in dom(a)}\sum_{k\in dom(c)} m_{0,i}\,n_{0,k}\right]$$

$$= \frac{m_{0,i}\,n_{0,k}}{Z} = P(a=i, c=k)$$

Where the term in the square brackets is ~~even~~ equal to 1 because it is equal to

$$\sum_a \sum_c P(a,c),$$ which is 1.

## Part 4

Let $M = \begin{bmatrix} \vec{m}_1 & \vec{m}_2 & \vec{m}_3 \end{bmatrix} = \begin{bmatrix} \vec{m}_1 & \vec{0} & \vec{0} \end{bmatrix} + \begin{bmatrix} \vec{0} & \vec{m}_2 & \vec{0} \end{bmatrix} + \begin{bmatrix} \vec{0} & \vec{0} & \vec{m}_3 \end{bmatrix}$

Let $N^T = \begin{bmatrix} \vec{n}_1^T \\ \vec{n}_2^T \\ \vec{n}_3^T \end{bmatrix} = \begin{bmatrix} \vec{n}_1^T \\ \vec{0}^T \\ \vec{0}^T \end{bmatrix} + \begin{bmatrix} \vec{0}^T \\ \vec{n}_2^T \\ \vec{0}^T \end{bmatrix} + \begin{bmatrix} \vec{0}^T \\ \vec{0}^T \\ \vec{n}_3^T \end{bmatrix}$

Multiplying together the expanded expressions and using the distributive property of matrix multiplication over matrix addition, the "cross-product" terms disappear and we are left with:

$$MN^T = \vec{m}_1\vec{n}_1^T + \vec{m}_2\vec{n}_2^T + \vec{m}_3\vec{n}_3^T .$$

## Part 5

Substitute $\vec{m}_2 = \lambda\vec{m}_1$ and $\vec{n}_3 = \gamma(\vec{n}_1 + \lambda\vec{n}_2)$ into the result of part 4 gives:

$$MN^T = \vec{m}_1\vec{n}_1^T + \lambda\vec{m}_1\vec{n}_2^T + \vec{m}_3\gamma(\vec{n}_1^T + \lambda\vec{n}_2^T)$$

$$= (\vec{m}_1 + \gamma\vec{m}_3)(\vec{n}_1 + \lambda\vec{n}_2)^T$$

which is of the form required.

```
%%Setup
clear all
import brml.*

%%Setup the free vectors with arbitrary values
m1 = [1; 2];  lamda = 3;  m3 = [2; 2];
n1 = [1; 1];  gamma = 2;  n2 = [1; 3];

%%Calculate the constrained vectors
m2 = lamda * m1;
n3 = gamma * (n1 + lamda * n2);

m0 = m1 + gamma * m3;
n0 = n1 + lamda * n2;

%%Create the M and N matrices
M = [m1 m2 m3];
N = [n1 n2 n3];

%%Populate the (unnormalised) joint probability distribution
distn_abc_unnorm =[];

for i_a = 1:2
   for i_b = 1:3
      for i_c = 1:2

          distn_abc_unnorm(i_a, i_b, i_c) = M(i_a, i_b) * N(i_c, i_b);

      end
   end
end

distn_abc_norm = distn_abc_unnorm / sum(distn_abc_unnorm(:));

%%Create jointpots array directly
a = 1;
b = 2;
c = 3;

jointpot = brml.array;
jointpot.variables = [1 2 3];
jointpot.table = distn_abc_norm;

%%Perform some sanity checks on the joint distribution
epsilon = 1e-15;

% Test 1
table_value = setpot(jointpot, [a, b, c], [1, 1, 1]);
hand_calculated_value = 1/154;
assert(table_value.table - hand_calculated_value < epsilon, 'Test 1
 failed');
```

```
% Test 2
table_value = setpot(jointpot, [a, b, c], [2, 3, 2]);
hand_calculated_value = 40/154;
assert(table_value.table - hand_calculated_value < epsilon, 'Test 2
 failed');


%%Calculate marginal distribution of a using brml toolbox
a_marginal = sumpot(jointpot, a, 0);  % sum over everything except a
a_marginal_vector = a_marginal.table;

c_marginal = sumpot(jointpot, c, 0);  % sum over everything except b
c_marginal_vector = c_marginal.table;

a_c_marginal = sumpot(jointpot, [a c], 0); % sum over everything
 except a and c

% Compare the joint distribution of a and c with the product of the
% individual distributions.
disp('Difference between P(A,B) and P(A)P(B):');
differences = a_marginal_vector * c_marginal_vector' -
 a_c_marginal.table
```

```
Difference between P(A,B) and P(A)P(B):

differences =

   1.0e-16 *

         0         0
   -0.2776   -0.5551
```

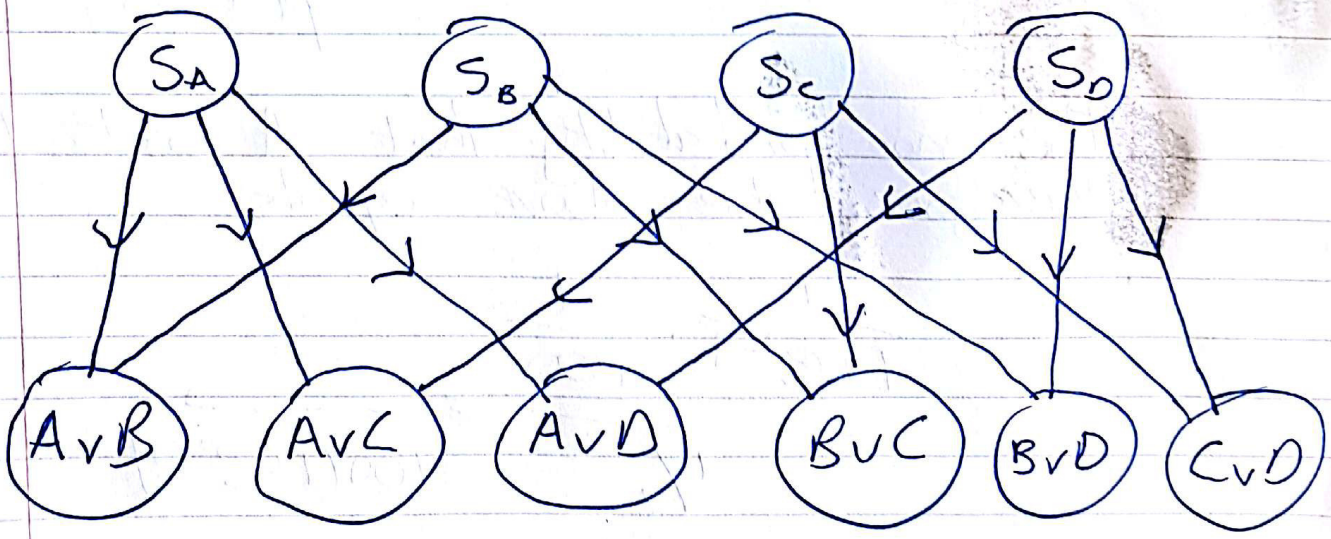
*Published with MATLAB® R2016b*

Exercise 3.21

Let $S_A \ldots S_D$ be the skill levels of the players (note that these are never observed, as in HMM)

The relevant independence assumptions are

1) the game is 2 player
2) a priori skill levels are independent
3) the outcome of all games given the skill levels (of the two players – see (2)) are independent




Each of the AvB, AvC ... is the set of results between the relevant two players

It can be seen that $S_A \not\perp\!\!\!\perp S_B$ but $S_A \perp\!\!\!\perp S_D \mid AvB$ since all other $X_r Y$

nodes are colliders, blocking any other link between A, B

⇒ the skill levels of the players are independent a posteriori

The prior belief is $p(S_A = 1, 2 \ldots 10) = \frac{1}{10}$

$p(S_B = 1, 2 \ldots 10) = \frac{1}{10}$

If A beats B the posterior skill probs are

$$p(S_A = i, S_B = j \mid A \text{ beat } B) = \frac{p(A \text{ beat } B \mid S_A = i, S_B = j)\, p(S_A = i, S_B = j)}{p(A \text{ beat } B)}$$

The prior probability that A beats B is $1/2$ therefore the above equals

$$\frac{\frac{1}{1+e^{i-j}} \cdot \frac{1}{100}}{1/2} = \frac{2}{100(1+e^{i-j})}$$

If A and B play again

Q.3.22.

**Question**:
- 10 Adverts, $A \in \{1 \dots 10\}$ at any given instance
  3 adds chosen at random
- Each add has a given interest level $S \in \{1 \dots 5\}$
- At each draw will refer to the adds chosen as A, B, C
- $P(A \text{ win} \mid S_A, S_B, S_C) \propto \exp(S_A - \max(S_B, S_C))$
- Initial interest levels are independent & uniform

**Idea**:
- Construct probability distribution table with 5 colums corresponding to (S) interest levels and 10 rows corresponding to the 10 different add choices.
- Initialise this with uniform probability of 0.2 in every cell (sum up to 1 row wise)
- Construct probability model to update 'belief' matrix after each instance of adds has been analysed.
- 20 instances of add choices are provided, therefore we will update the orginal matrix 20 times, each time populating 3 distinct rows of this matrix.
- The final matrix should yield the most likely probability distribution for the interest levels (S)

Pseudo Code:

```
for timestep 1:20
  initialise matrices M(t) & M(t+1)
for rows [row A, row B, row C]
  for col. 1:5
      % calc. for each element
        the numerator (refer to model)
        (2 nested summations present)
      % calc. the denominator (refer to model)
        -3 nested summations
      % write to new matrix Mt+1 for each (row,col) value: num/denom
  end
end
M(t) = M(t+1)   % update matrices to next timestep
end
```

# MODEL

$$P(S_A \mid A \text{ wins } B,C) = \frac{\sum_j^{\text{col.}} \sum_i^{\text{rows}} P(A \text{ win } B,C \mid S_A,S_B,S_C)\, P(S_A \mid B,C)\, P(B \mid C)\, P(C)}{\sum_k \sum_j^{\text{col.}} \sum_i^{\text{rows}} P(A \text{ win } B,C \mid S_A,S_B,S_C)\, P(S_A \mid B,C)\, P(B \mid C)\, P(C)}$$

- numerator: fixed new time step matrix
- denominator: use old time step info.
- $P(A \text{ win } B,C \mid S_A,S_B,S_C)$: ①
- $P(S_A \mid B,C)\, P(B \mid C)\, P(C)$: ②

① know that:

$$P(A \text{ wins } B,C \mid S_A,S_B,S_C) \propto e^{(S_A - \max(S_B,S_C))}$$

can be normalised by summing over all possible values of the exp. function to give:

$$P(A \text{ win } B,C \mid S_A,S_B,S_C) = \frac{e^{(S_A - \max(S_B,S_C))}}{e^{(S_A - \max(S_B,S_C))} + e^{(S_B - \max(S_A,S_C))} + e^{(S_C - \max(S_A,S_B))}}$$

↑ will call this Myexp. function in code.

② Assuming independence the expression can be reduced to: $P(S_A \mid B,C)\, P(B \mid C)\, P(C) = P(A)\, P(B)\, P(C)$

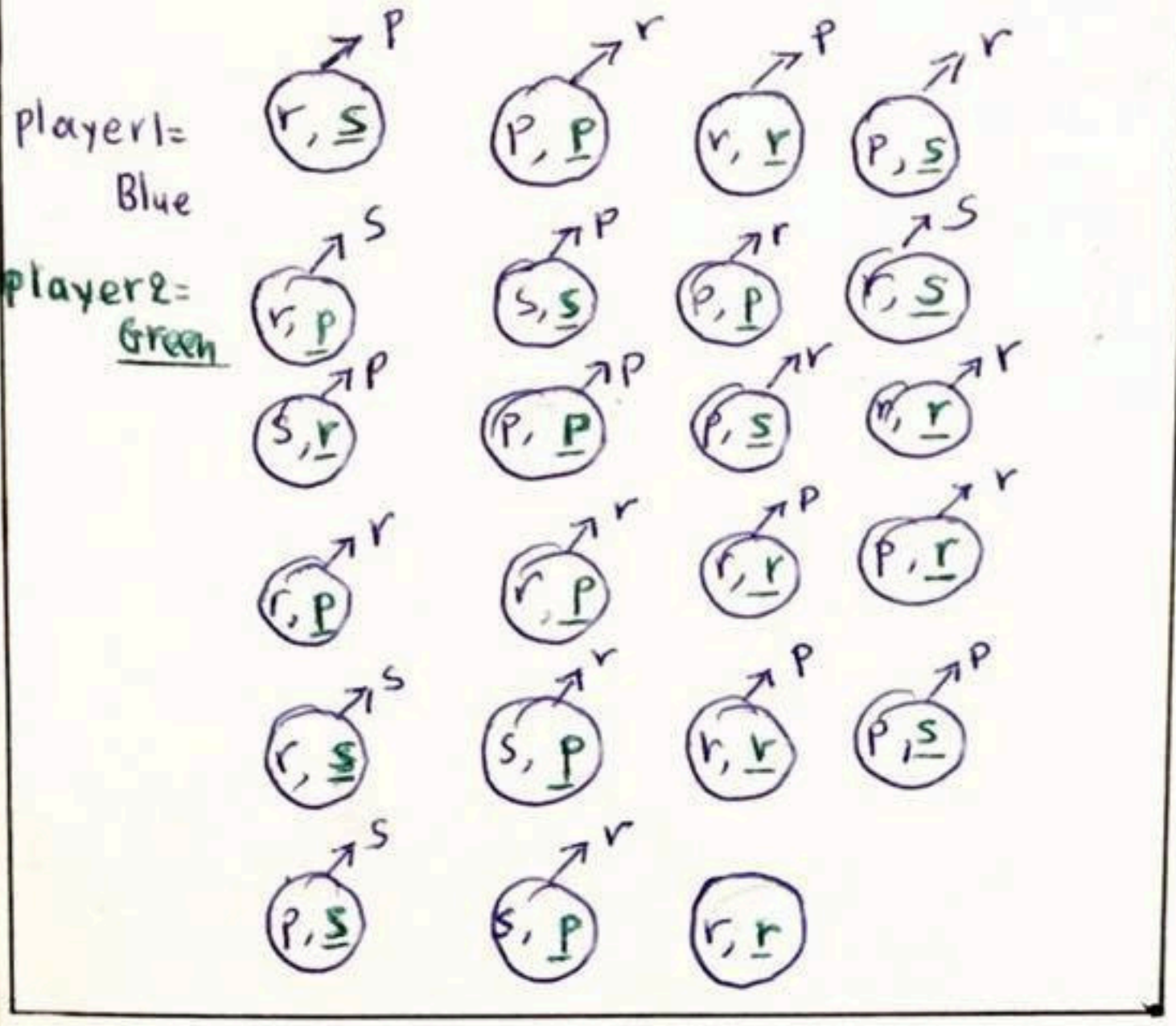
```
% the matrix int_levels(i,j) is the probability that the interest
 level in the ith
% ad is equal to j. int_levels(i,j) is therefore a 10x5 matrix with
 all entries
% initially set to 0.2 as per the problem

int_levels(1:10, 1:5)=0.2;

assert(all(sum(int_levels,2) == ones(10,1)), 'initial probabilities do
 not sum to one');

% the Daily Fail website data is stored as an array of 20 rows, each
 row
% holds the three entries denoting the preferences each viewer
 expresses

P=[1,2,3; 2,5,3; 4,7,10; 6,3,4; 6,8,5; 9,3,7; 10,2,4; 7,1,2; 8,7,2;
 8,3,6;...
    8,6,4; 4,3,9; 5,4,1; 9,5,1; 6,7,8; 4,9,7; 10,8,6; 5,4,3; 6,3,2;
 1,4,2];


assert(all(size(P)==[20,3]), 'input preference data not as
 specified');

for i = 1:20
    %iterating through the dataset
    row_pref = P(i,:);
    a=row_pref(1);
    b=row_pref(2);
    c=row_pref(3);
    % compute the prior probability of the user choosing a given a,b,c
    evidence=prob_choose_a(a,b,c, int_levels);
    for j = 1:5
        % update the probabilities of the interest levels of a using
        % Bayes rule, given the prior, the evidence. the likelihood of
 a
        % being chosen
        new_int_levels(a,j)=int_levels(a,j)*prob_choose_a_given_int(j,
 b, c, int_levels)/evidence;
    end
    for j= 1:5
        % update the probabilities of the interest levels of b using
        % Bayes rule, given the prior (on b), the evidence (same) and
 the
        % likelihood of a being chosen given b's interest level is j

 new_int_levels(b,j)=int_levels(b,j)*prob_choose_a_given_b_int(a,
 j ,c, int_levels)/evidence;
    end
    for j= 1:5
        %update probabilities for c, essentially the same as b
```

```
 new_int_levels(c,j)=int_levels(c,j)*prob_choose_a_given_b_int(a, j,
 b, int_levels)/evidence;
    end
    int_levels(a,:)=new_int_levels(a,:);
    int_levels(b,:)=new_int_levels(b,:);
    int_levels(c,:)=new_int_levels(c,:);
end
%display matrix rounded to 3 decimal places for display
printmat(round(new_int_levels,3), 'Expected interest levels', 'Add1
 Add2 Add3 Add4 Add5 Add6 Add7 Add8 Add9 Add10', 'S1 S2 S3 S4 S5')


function total_prob=prob_choose_a(a,b,c, int_levels)
% calculates the probability that, served with ads a, b, c, the user
 will
% pick a

if any([a==b, b==c, a==c])
    fprintf('error')
end

total_prob=0;

% sums the probabilities that ad a will be chosen, given that its
 interest
% level is i


for i = 1:5
    total_prob=total_prob+int_levels(a,i)*prob_choose_a_given_int(i,
 b, c, int_levels);
end

end



function total_prob=prob_choose_a_given_int(i,b,c, int_levels)
% calculates the probability that, served with an ad of interest level
 i
% and ads b,c the user will pick the ad of interest level i

total_prob=0;

for j = 1:5
    for k= 1:5
        % compute the joint probability that the interest levels in
 ads
        % b,c are j and k
        joint_prob=int_levels(b,j)*int_levels(c,k);
        % compute the probability that the user then clicks A
        prob_click=prob_int(i,j,k);
        prob_click_A=prob_click(1);
```

```
        total_prob=total_prob+joint_prob*prob_click_A;
    end
end


end

function total_prob=prob_choose_a_given_b_int(a, j, c, int_levels)
% calculates the probability that, served with an ad a, and ad b of
% interest j and an ad c, the user will choose a

total_prob=0;

for i = 1:5
    for k= 1:5
        % compute the joint probability that the interest levels in
 ads
        % a,c are j and k
        joint_prob=int_levels(a,i)*int_levels(c,k);
        % compute the probability that the user then clicks A
        prob_click=prob_int(i,j,k);
        prob_click_A=prob_click(1);
        total_prob=total_prob+joint_prob*prob_click_A;
    end
end


end




function [prob_click_A prob_click_B prob_click_C]=prob_int(i,j,k)

% computes the probability that the user will click A, B, C given a
 choice
% of ads with interest level i,j,k

pot_A=exp(i-max(j,k));
pot_B=exp(j-max(i,k));
pot_C=exp(k-max(i,j));

sum_pots=pot_A+pot_B+pot_C;

prob_click_A=pot_A/sum_pots;
prob_click_B=pot_B/sum_pots;
prob_click_C=pot_C/sum_pots;
total_prob=prob_click_A+prob_click_B+prob_click_C;

assert(all([0.99<total_prob, 1.01>total_prob]), 'probabilities derived
 from potential do not sum to one')

end
```

```
Expected interest levels =
                        S1           S2           S3           S4
      S5
         Add1       0.01800      0.11500      0.44300      0.39000
 0.03500
         Add2       0.19700      0.40000      0.35200      0.05100
       0
         Add3       0.54800      0.33900      0.10600      0.00700
       0
         Add4       0.01100      0.15300      0.61800      0.21700
 0.00200
         Add5       0.00800      0.05700      0.32100      0.51800
 0.09700
         Add6             0      0.00100      0.04500      0.45600
 0.49800
         Add7       0.16100      0.31100      0.37300      0.14900
 0.00700
         Add8             0      0.00200      0.05300      0.49300
 0.45200
         Add9       0.02300      0.14600      0.43600      0.34400
 0.05200
        Add10       0.00200      0.01100      0.06500      0.27300
 0.64900
```

*Published with MATLAB® R2016b*




Set r ; for rock
s ; for scissors
p ; for paper

and let player 1 play first.

figure 1: In cycles we can see the previous moves of player 1 and player 2 and with the arrows we indicate player's 1 next move

| $(x_t^1, x_t^2)$ / $x_{t+1}^1$ | (r,s) | (s,r) | (p,r) | (r,p) | (p,r) | (p,p) | (s,s) | (s,p) | (r,r) | (p,s) |
|---|---|---|---|---|---|---|---|---|---|---|
| r | | | I | II | II | | II | II | I | II |
| p | I | I | | | I | I | | | III | I |
| s | II | | | I | | | | | | I |

Table 1: Table of move counts, where $x_t^1$ indicates the move of player 1 at time step t, similarly $x_t^2$ indicates the move of player 2 at time step t, and $x_{t+1}^1$ the move of player 1 at time step t+1.

| $(x_t^1, x_t^2)$ / $x_{t+1}^1$ | (r,s) | (s,r) | (p,r) | (r,p) | (p,r) | (p,p) | (s,s) | (s,p) | (r,r) | (p,s) |
|---|---|---|---|---|---|---|---|---|---|---|
| r | 0 | 0 | $1/22$ | $2/22$ | $2/22$ | 0 | $2/22$ | $2/22$ | $1/22$ | $2/22$ |
| p | $1/2$ | $1/22$ | 0 | 0 | $1/22$ | $1/22$ | 0 | 0 | $3/22$ | $1/22$ |
| s | $2/22$ | 0 | 0 | $1/22$ | 0 | 0 | 0 | 0 | 0 | $1/22$ |

Table 2: Table of frequencies for each move - (same labeling as Table 1).

1. Assuming that player 1 plays according to a first order Markov Chain as it defined in (3.7.15) we can write;

$$P\left(X_{2:T}^{1} \mid X_{1:T-1}^{1}, X_{1:T-1}^{2}\right) = P\left(X_2^1, X_3^1, \cdots, X_{23}^1 \mid x_1^1, x_2^1, \cdots, x_{22}^1, x_1^2, x_2^2, \cdots, x_{22}^2\right) =$$

$$= P\left(X_{23}^1 \mid X_{22}^1, X_{22}^2\right) \overset{(*)}{=} P\left(X_{t+1}^1 = r \mid X_{22}^2 = p, X_{22}^1 = s\right)$$

$$\overset{(**)}{=} 2/22$$

(*) based on figure 1.

(**) based on Table 2.

2.

| | Wins | Draws |
|---|---|---|
| player 1 | 6/22 | 7/22 |
| player 2 | 9/22 | |

Table 3: Wins and Draws for each player

Assume that at each step Player 1 choose the next move randomly, then: $P(\text{Player 1} = \text{Win}) = \frac{1}{3}$, $P(\text{Player 1} = \text{Draw}) = \frac{1}{3}$ and $P(\text{Player 1} = \text{lose}) = \frac{1}{3}$. According to Table 3, Player 1 wins 6 games out of 22. Hence the probability that Player 1 wins 6 games out of 22 given that is playing random moves is given by;

$$P(\text{Player 1 wins 6 games out of 22} \mid \text{random stategy}) = \binom{22}{6}\left(\frac{1}{3}\right)^6\left(\frac{2}{3}\right)^{16} = 0.1558$$

Hence we can say that is much more unlikely that player 1 plays random moves.

3. for calculating the probability that Player 1 plays each or rock, paper, scissors at the next time step we can base on Table 2 and conclude that;

$$P(X^1_{T+1} = r \mid X^1_T = r,\ X^2_T = r) \overset{(*)}{=} 1/22$$

$$P(X^1_{T+1} = p \mid X^1_T = r,\ X^2_T = r) \overset{(*)}{=} 3/22$$

$$P(X^1_{T+1} = s \mid X^1_T = r,\ X^2_T = r) \overset{(*)}{=} 0$$

(*) from figure 1 we can see that the last moves of Player 1 and Prayer 2 at time step T are;

$$(X^1_T, X^2_T) = (r, r)$$

4. From question 3 we can see that the most "likely" move of player 1 at time step T+1 given the immediate previous moves at time step T of player 1 and player 2 is paper. Hence the move for player 2 considering the rules of the game is Scissors.

## Exercise 3.24

1) Using terms from equation (3.7.17), in general

$p(x_1, y_1)$ can be written as $p(x_1)\,p(y_1|x_1)$ or $P(y_1)P(x_1|y_1)$

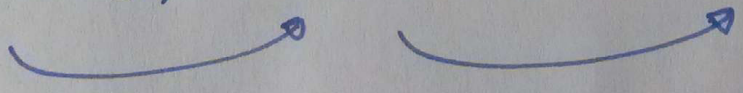
$p(x_2, y_2|x_1, y_1)$ can be written as $P(y_2|x_1, y_1)P(x_2|x_1, y_1, y_2)$

$p(x_3, y_3|x_2, y_2)$ can be written as $P(y_3|x_2, y_2)P(x_3|x_2, y_2, y_3)$

---

Compare the following:
- $P(x_1, y_1, x_2, y_2, x_3, y_3)$ written out in cascade form with variable order $y_1, x_1, y_2, x_2, y_3, x_3$
- Eqn (3.7.17) with the substitutions above.
- The expression for $P_1(x_1, y_1, x_2, y_2, x_3, y_3)$ in (3.7.16)

| Cascade | Eqn 3.7.17 | Eqn 3.7.16 |
|---|---|---|
| $p(y_1)$ | $p(y_1)$ | $p(y_1)$ |
| $\times p(x_1\|y_1)$ | $\times p(x_1\|y_1)$ | $\times p(x_1\|y_1)$ |
| $\times p(y_2\|x_1, y_1)$ | $\times p(y_2\|x_1, y_1)$ | $\times p(y_2)$ |
| $\times p(x_2\|x_1, y_1, y_2)$ | $\times p(x_2\|x_1, y_1, y_2)$ | $\times p(x_2\|x_1, y_2)$ |
| $\times p(y_3\|x_1, y_1, x_2, y_2)$ | $\times p(y_3\|x_2, y_2)$ | $\times p(y_3)$ |
| $\times p(x_3\|x_1, y_1, x_2, y_2, y_3)$ | $\times p(x_3\|x_2, y_2, y_3)$ | $\times p(x_3\|x_2, y_3)$ |



where I have underlined in red the conditional variables that have been removed as you move between columns.

Because you can represent any distribution in cascade form without loss of generality, arriving at (3.7.17) and then (3.7.16) represents two sets of conditional independence assumptions.

So the expression on the previous page for (3.7.17) can be seen to be a 'generalisation' of (3.7.16). By making the conditional independence assumptions implicit in (3.7.16) we arrive at equality.

ie:

$$(3.7.17) \rightarrow p(x_1, y_1)\, p(x_2, y_2 \mid x_1, y_1)\, p(x_3, y_3 \mid x_2, y_2)$$

$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid \underline{x_1, y_1})\, p(x_2 \mid x_1, \underline{y_1, y_2})\, p(y_3 \mid \underline{x_2, y_2})\, p(x_3 \mid x_2, \underline{y_2, y_3})$$

Information in (3.7.16) allows us to make the conditional independence assumptions to remove the terms underlined in red.

The resulting expression then is (3.7.16).

2) Already shown in part (i) that distribution $p_1$ can be written as in (3.7.17).

So now show that (3.7.17) and (3.7.18) are equivalent.

(3.7.17): $p(x_1, y_1)\, p(x_2, y_2 \mid x_1, y_1)\, p(x_3, y_3 \mid x_2, y_2)$

expandout using the definition of conditional probability

$$= p(x_1, y_1)\, \frac{p(x_1, y_1, x_2, y_2)}{p(x_1, y_1)}\, \frac{p(x_2, y_2, x_3, y_3)}{p(x_2, y_2)}$$

By swapping the terms in the denominator and using the definition of conditional probability again, we get

~~$p(x_1, y_1)\, p($~~

Cancel the $p(x_1, y_1)$ terms and multiply numerator and denominator by $p(x_3, y_3)$ to get:

$$p(x_3, y_3)\, \frac{p(x_2, y_2, x_3, y_3)}{p(x_3, y_3)}\, \frac{p(x_1, y_1, x_2, y_2)}{p(x_2, y_2)}$$

Apply the definition of conditional probability to get

$p(x_3, y_3)\, p(x_2, y_2 \mid x_3, y_3)\, p(x_1, y_1 \mid x_2, y_2)$, which is (3.7.18)

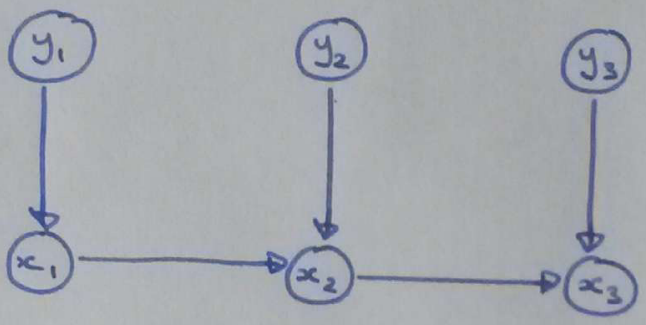
3) Distribution $P_2$ can be expanded to the following Belief Net:

$$P(y_3)\,P(x_3|y_3)\,P(y_2|x_3,y_3)\,P(x_2|x_3,y_3,y_2)\,P(y_1|x_2,y_2)\,P(x_1|x_2,y_2,y_1)$$

Expression (3.7.16) does not allow us to drop any of these terms.

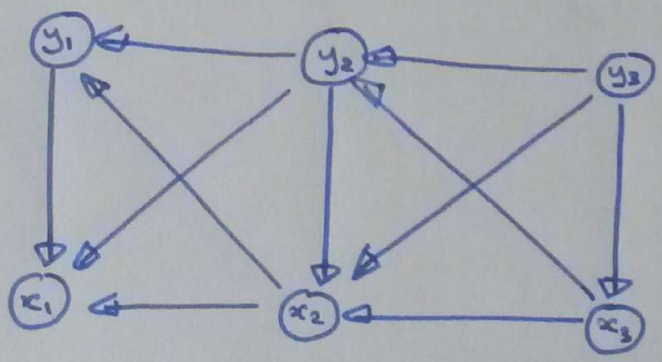
So, drawing out the DAGs associated with $P_1$ and $P_2$:

## P1




Total edges: 5

## P2

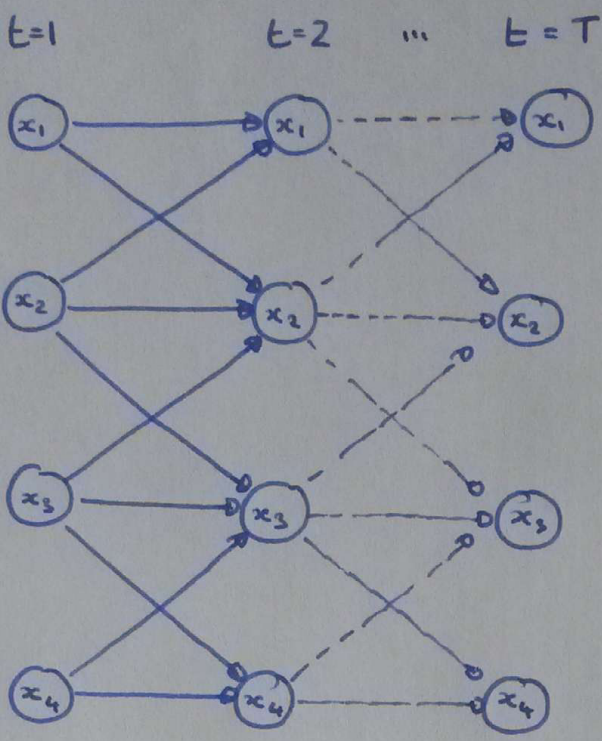



Total edges: 11

So DAG for distribution for P1 has fewer edges.

# Exercise 3.25

1)




10 edges in every space between rows.

(T-1) spaces ⟶ total of $10 \times (T-1)$ edges.

2) To do this we will show that the "time reversed" form can be manipulated into a causal form. We will then apply the "physical" conditional independence assumptions to arrive at Eq. (3.7-19)

---

(3.7.20) gives $p(\vec{x}_T) \prod_{t=1}^{T-1} p(\vec{x}_t \mid \vec{x}_{t+1})$

Apply Bayes' rule to give:

$$p(\vec{x}_T) \prod_{t=1}^{T-1} \frac{p(\vec{x}_{t+1} \mid \vec{x}_t)\, p(\vec{x}_t)}{p(\vec{x}_{t+1})}$$

Change the index range to $t = 2, \ldots, T$, giving

$$p(\vec{x}_T) \prod_{t=2}^{T} \frac{p(\vec{x}_t \mid \vec{x}_{t-1})\, p(\vec{x}_{t-1})}{p(\vec{x}_t)}$$

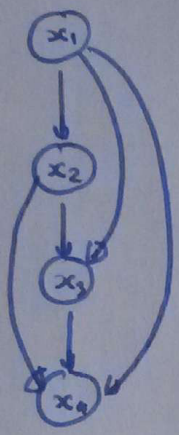
Recognising that most of the $p(\vec{x}_t)$ and $p(\vec{x}_{t-1})$ terms cancel when this product is expanded, due to the 'telescope' effect; we get

$$p(\vec{x}_1) \prod_{t=2}^{T} p(\vec{x}_t \mid \vec{x}_{t-1})$$

This is of the form in (3.7.19), just without the "physical" conditional independence assumptions made.

Applying these assumptions therefore arrives at (3.7.19)

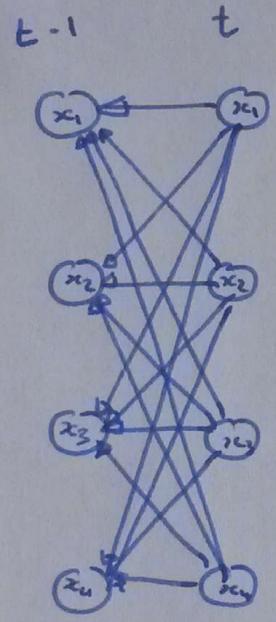
Without making the conditional independence assumptions, $p(\vec{x}_t)$ can be written in cascade form, eg




total edges: 6

Those assumptions can't be made because they are expressed in (3.7.19) in the wrong form.

And each timestep depends on all its "previous" (anti-causal!) timestep:




total edges: $4 \times 4 = 16$

Combining these two into a DAG with $T$ timesteps (and $T-1$ "gaps") has total edges:

$$\underbrace{6 \times T}_{\text{from top diagram}} + \underbrace{16 \times (T-1)}_{\text{from bottom diagram.}} = 22 \times T - 16$$

(c.f. $10 \times (T-1)$ from Part 1).