

## Graphical Models Coursework 3

Razvan Valentin Marinescu	Konstantinos Georgiadis
Student Number: 14060166	Student Number: 14110861
<code>razvan.marinescu.14@ucl.ac.uk</code>	<code>konstantinos.georgiadis.14@ucl.ac.uk</code>

December 11, 2014

Just as in the previous assignment, we both did the exercises independently and compared our answers until we agreed on what to choose. Konstantinos wrote most of the report and Razvan added some changes afterwards.

## Exercise 7.4

We changed the code from demoMDP accordingly for this exercise and we only kept the Value Iteration strategy. We kept the value of gamma to be the same as it was (0.95), as well as the number of iterations to be 30. Our code then proceeds to make moves by moving at each timestep to the neighboring grid position with the highest value and stops once it reaches a point where the grid position that the airplane currently occupies has a higher value than its neighboring grid positions. For the second part of the exercise, we merely had to change the p matrix accordingly. The optimal paths are given from the variable PositionSequence. In the first case, the optimal path is:

```
X Y
1 13
1 12
1 11
2 11
3 11
4 11
5 11
6 11
7 11
8 11
9 11
10 11
11 11
12 11
13 11
14 11
15 11
15 10
15 9
14 9
14 8
14 7
13 7
12 7
11 7
10 7
9 7
8 7
7 7
6 7
5 7
4 7
4 6
4 5
4 4
```

5 4  
6 4  
7 4  
8 4

In the second case, the optimal path is:

X Y  
1 13  
1 14  
2 14  
3 14  
4 14  
5 14  
6 14  
7 14  
8 14  
9 14  
10 14  
11 14  
12 14  
13 14  
14 14  
15 14  
16 14  
16 13  
16 12  
16 11  
15 11  
15 10  
15 9  
14 9  
14 8  
14 7  
13 7  
12 7  
11 7  
10 7  
9 7  
8 7  
7 7  
6 7  
5 7  
4 7  
4 6  
4 5  
4 4  
5 4  
6 4  
7 4

This result can be explained in the sense that the value iteration method makes use of the  $p$  matrix, as well as the utility values. Since there is now a higher chance to move up, during the value iterations, the values of the path around the up-right village get higher values, rather than the path through (14,11).

MATLAB code:

```

1 clear all;
2 close all;
3 load('airplane.mat');
4 import brml.*
5 [Gx, Gy] = size(U); % two dimensional grid size
6 S = Gx*Gy; % number of states on grid
7 st = reshape(1:S,Gx,Gy); % assign each grid point a state

9 A = 5; % number of action (decision) states
[stay, up, down, left, right] = assign(1:A); % actions (decisions)
11 p = zeros(S,S,A); % initialise the transition p(xt|xtm,dtm) ie p(x(t)|x(t-1),d(t-1))

13 % make a deterministic transition matrix on a 2D grid:
for x = 1:Gx
15     for y = 1:Gy
16         p(st(x,y),st(x,y),stay)=1; % can stay in same state
17         if validgridposition(x+1,y,Gx,Gy)
18             p(st(x+1,y),st(x,y),right)=1;
19         end
20         if validgridposition(x-1,y,Gx,Gy)
21             p(st(x-1,y),st(x,y),left)=1;
22         end
23         if validgridposition(x,y+1,Gx,Gy)
24             p(st(x,y+1),st(x,y),up)=1;
25         end
26         if validgridposition(x,y-1,Gx,Gy)
27             p(st(x,y-1),st(x,y),down)=1;
28         end
29     end
30 end
31 % define utilities
u = U(:);
33 gamma = 0.95; % discount factor
figure; imagesc(reshape(u,Gx,Gy)); colorbar; title('utilities');
35 [xt, xtm, dtm]=assign(1:3); % assign the variables x(t), x(t-1), d(t-1) to some numbers

37 % define the transition potentials p(x(t)|x(t-1),d(t-1))
tranpot=array([xt xtm dtm],p);
39 % setup the value potential v(x(t))
valpot=array(xt,ones(S,1)); % initial values

41 maxiterations=30; tol=0.001; % termination criteria
43 % Value Iteration:
oldvalue=valpot.table;
45 for valueloop=1:maxiterations
46     valueloop
47     tmpptot = maxpot(sumpot(multipots([tranpot valpot]),xt),dtm);
48     valpot.table = u + gamma*tmpptot.table; % Bellman's recursion
49     if mean(abs(valpot.table-oldvalue))<tol; break; end % stop if converged
oldvalue = valpot.table;

```

```

51  imagesc(reshape(valpot.table,Gx,Gy)); colorbar; drawnow
end
53  figure; bar3zcolor(reshape(valpot.table,Gx,Gy));

55  FinalValues = reshape(valpot.table,Gx,Gy);

57  %Calculate Optimal Sequence
PositionSequence = [1 13];
59  timestep = 1;
while(1)
61      x = PositionSequence(timestep,1);
        y = PositionSequence(timestep,2);
63      %find best move
        currentval = FinalValues(x,y);
65      bestmove = stay;
        if validgridposition(x+1,y,Gx,Gy)
67          if(FinalValues(x+1,y) > currentval)
                currentval = FinalValues(x+1,y);
69          bestmove = right;
            end
71      end
        if validgridposition(x-1,y,Gx,Gy)
73          if(FinalValues(x-1,y) > currentval)
                currentval = FinalValues(x-1,y);
75          bestmove = left;
            end
77      end
        if validgridposition(x,y+1,Gx,Gy)
79          if(FinalValues(x,y+1) > currentval)
                currentval = FinalValues(x,y+1);
81          bestmove = up;
            end
83      end
        if validgridposition(x,y-1,Gx,Gy)
85          if(FinalValues(x,y-1) > currentval)
                currentval = FinalValues(x,y-1);
87          bestmove = down;
            end
89      end
        %make new move or exit
91      timestep = timestep + 1;
        if(bestmove == stay)
93          break;
        end
95      PositionSequence(timestep,1:2) = PositionSequence(timestep-1,1:2);
        if(bestmove == right)
97          PositionSequence(timestep,1) = PositionSequence(timestep,1)+1;
        end
99      if(bestmove == left)
        PositionSequence(timestep,1) = PositionSequence(timestep,1)-1;
101     end
        if(bestmove == up)
103         PositionSequence(timestep,2) = PositionSequence(timestep,2)+1;
        end
105     if(bestmove == down)
        PositionSequence(timestep,2) = PositionSequence(timestep,2)-1;
107     end
end
109 PositionSequence
111
113 %Part 2

```

```

115 p = zeros(S,S,A);
116 for x = 1:Gx
117     for y = 1:Gy
118         p(st(x,y),st(x,y),stay)=1;
119         if validgridposition(x+1,y,Gx,Gy)
120             if validgridposition(x,y+1,Gx,Gy)
121                 p(st(x+1,y),st(x,y),right)=0.9;
122                 p(st(x,y+1),st(x,y),right)=0.1;
123             else
124                 p(st(x+1,y),st(x,y),right)=1;
125             end
126         if validgridposition(x-1,y,Gx,Gy)
127             p(st(x-1,y),st(x,y),left)=1;
128         end
129         if validgridposition(x,y+1,Gx,Gy)
130             p(st(x,y+1),st(x,y),up)=1;
131         end
132         if validgridposition(x,y-1,Gx,Gy)
133             p(st(x,y-1),st(x,y),down)=1;
134         end
135     end
136 end
137 % define utilities
138 u = U(:);
139 gamma = 0.95;
140 figure; imagesc(reshape(u,Gx,Gy)); colorbar; title('utilities');
141 [xt, xtm, dtm]=assign(1:3); % assign the variables x(t), x(t-1), d(t-1) to some
    numbers
142
143 % define the transition potentials p(x(t)|x(t-1),d(t-1))
144 tranpot=array([xt xtm dtm],p);
145 % setup the value potential v(x(t))
146 valpot=array(xt,ones(S,1)); % initial values
147
148 maxiterations=30; tol=0.001; % termination criteria
149 % Value Iteration:
150 oldvalue=valpot.table;
151 for valueloop=1:maxiterations
152     valueloop
153     tmppot = maxpot(sumpot(multipots([tranpot valpot]),xt),dtm);
154     valpot.table = u + gamma*tmppot.table; % Bellman's recursion
155     if mean(abs(valpot.table-oldvalue))<tol; break; end % stop if converged
156     oldvalue = valpot.table;
157     imagesc(reshape(valpot.table,Gx,Gy)); colorbar; drawnow
158 end
159 figure; bar3zcolor(reshape(valpot.table,Gx,Gy));
160
161 FinalValues = reshape(valpot.table,Gx,Gy);
162
163 %Calculate Optimal Sequence
164 PositionSequence = [1 13];
165 timestep = 1;
166 while(1)
167     x = PositionSequence(timestep,1);
168     y = PositionSequence(timestep,2);
169     %find best move
170     currentval = FinalValues(x,y);
171     bestmove = stay;
172     if validgridposition(x+1,y,Gx,Gy)
173         if (FinalValues(x+1,y) > currentval)
174             currentval = FinalValues(x+1,y);
175             bestmove = right;

```

```

177     end
178 end
179 if validgridposition(x-1,y,Gx,Gy)
180     if (FinalValues(x-1,y) > currentval)
181         currentval = FinalValues(x-1,y);
182         bestmove = left;
183     end
184 end
185 if validgridposition(x,y+1,Gx,Gy)
186     if (FinalValues(x,y+1) > currentval)
187         currentval = FinalValues(x,y+1);
188         bestmove = up;
189     end
190 end
191 if validgridposition(x,y-1,Gx,Gy)
192     if (FinalValues(x,y-1) > currentval)
193         currentval = FinalValues(x,y-1);
194         bestmove = down;
195     end
196 end
197 %make new move or exit
198 timestep = timestep + 1;
199 if(bestmove == stay)
200     break;
201 end
202 PositionSequence(timestep,1:2) = PositionSequence(timestep-1,1:2);
203 if(bestmove == right)
204     PositionSequence(timestep,1) = PositionSequence(timestep,1)+1;
205 end
206 if(bestmove == left)
207     PositionSequence(timestep,1) = PositionSequence(timestep,1)-1;
208 end
209 if(bestmove == up)
210     PositionSequence(timestep,2) = PositionSequence(timestep,2)+1;
211 end
212 if(bestmove == down)
213     PositionSequence(timestep,2) = PositionSequence(timestep,2)-1;
214 end
215 end
PositionSequence

```

## Exercise 7.13

MATLAB function (explanation below it):

```

1 function [d1, val]=optdec(epsilonA1,epsilonB1,desired,T,w1,pars)
2 import brml.*
3 eat = 1;
4 eatm1 = 2;
5 peatgeatm1=array([eat eatm1],pars.epsilonAtran);
6
7 ebt = 3;
8 ebtml = 4;
9 pebtgebtm1=array([ebt ebtml],pars.epsilonBtran);
10
11 wt = 5;
12 wtml = 6;
13 utilitywT = pars.WealthValue;
14 utilitywT(utilitywT < desired*w1) = 0;
15 utilitywT(utilitywT >= desired*w1) = 10000;

```

```

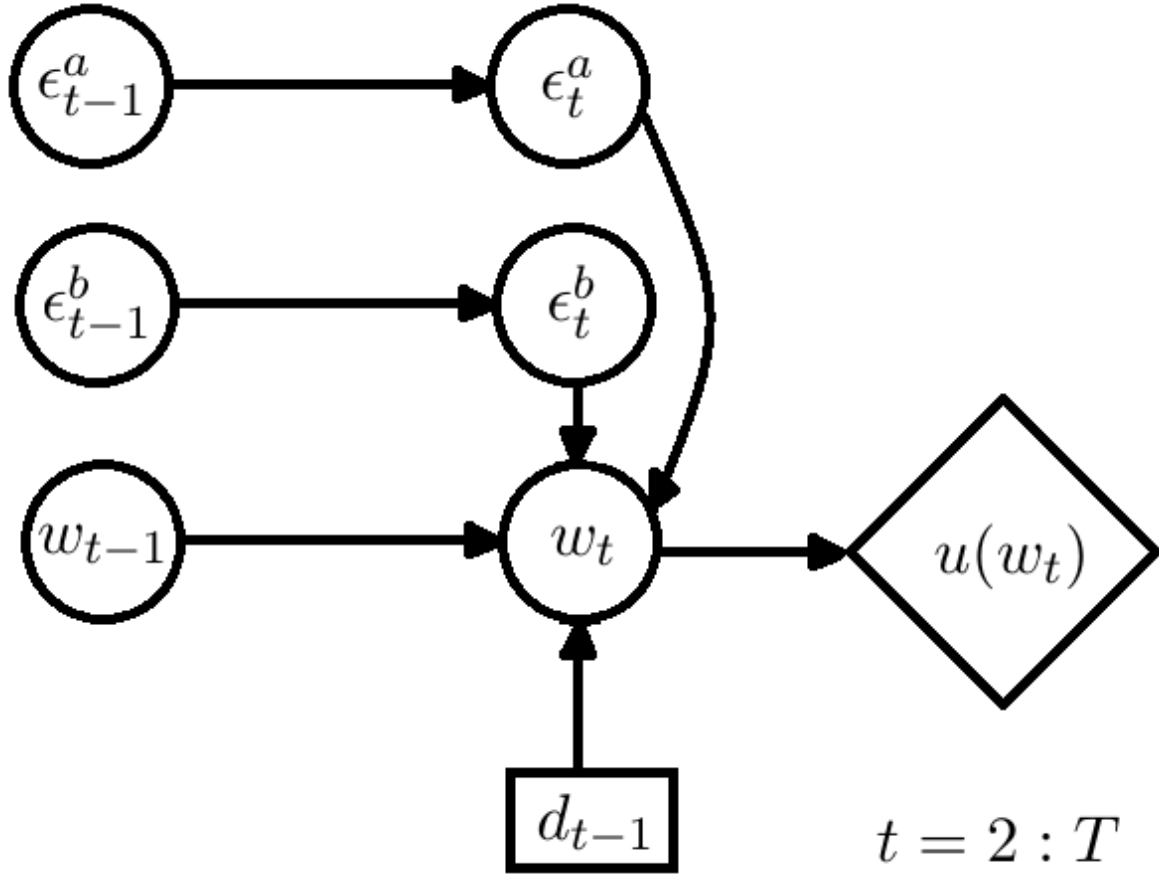
uwT=array(wt,utilitywT);
17 dtm1 = 7;
19 pbigarray = zeros(length(pars.WealthValue),length(pars.WealthValue),length(pars.
    epsilonAval),length(pars.epsilonBval),length(pars.DecisionValue));
21 for a = 1:length(pars.WealthValue)
    previouswealth = pars.WealthValue(a);
23     for b = 1:length(pars.epsilonAval)
        currentincreasea = pars.epsilonAval(b);
25         for c = 1:length(pars.epsilonBval)
            currentincreaseb = pars.epsilonBval(c);
27             for d = 1:length(pars.DecisionValue)
                previousdecision = pars.DecisionValue(d);
29                 newwealth = previouswealth*(previousdecision*(1+currentincreasea)
                    +(1-previousdecision)*(1+currentincreaseb));
                    %find closest value
31                 [~,diracposition] = min(abs(pars.WealthValue - newwealth));
                    pbigarray(diracposition,a,b,c,d) = 1;
33             end
        end
35     end
end
37 pbig=array([wt wtm1 eat ebt dtm1],pbigarray);

39 multiypots = multpots([peatgeatm1 pebtgebtm1 pbig]);
for t = T:-1:2
41     if(t == T)
        messagepot = maxpot(sumpot(multpots([multiypots uwT]),[eat ebt wt]),
            dtm1);
43     elseif(t > 2)
        messagepot.variables = [eat ebt wt];
45         messagepot = maxpot(sumpot(multpots([multiypots messagepot]),[eat ebt
            wt]),dtm1);
        else
47             messagepot.variables = [eat ebt wt];
            UtilityFinal = sumpot(multpots([multiypots messagepot]),[eat ebt wt]);
49         end
    end
51 UtilityFinalGivenConds = setpot(UtilityFinal,[eatm1 ebtm1 wtm1],[epsilonA1
    epsilonB1 find(wl==pars.WealthValue)]);
    [~, d1] = max(UtilityFinalGivenConds.table);
53 %Calculate Expected Wealth Value
55 wealthvalues=array(wt,pars.WealthValue);
for t = 2:1:T
57     if(t == T)
        messagepot.variables = [eatm1 ebtm1 wtm1];
59         FinalPot = sumpot(multpots([multiypots messagepot]),[eatm1 ebtm1 wtm1
            dtm1 eat ebt]);
            FinalPot.table = FinalPot.table/(length(pars.DecisionValue));
61     elseif(t > 2)
        messagepot.variables = [eatm1 ebtm1 wtm1];
63         messagepot = sumpot(multpots([multiypots messagepot]),[eatm1 ebtm1 wtm1
            dtm1]);
            messagepot.table = messagepot.table/(length(pars.DecisionValue));
65     else
        messagepot = setpot(multiypots,[eatm1 ebtm1 wtm1 dtm1],[epsilonA1
            epsilonB1 find(wl==pars.WealthValue) d1]);
67     end
end
69 ExpectedWealth = sumpot(multpots([FinalPot wealthvalues]));
val = ExpectedWealth.table;

```



After creating the probability tables for  $p(\epsilon_t^a | \epsilon_{t-1}^a)$ ,  $p(\epsilon_t^b | \epsilon_{t-1}^b)$ ,  $p(w_t | w_{t-1}, \epsilon_t^a, \epsilon_t^b, d_{t-1})$ , we perform the message passing as described in equations 7.8.34-36. The optimal expected utility is: 9875.5933608406 and the optimal choice is to invest 0.25 of our wealth into asset a at the first timestep. Later on, the code calculated the expected wealth at timestep T, even though the exercise doesn't seem to ask for it, but the file `exerciseInvest.m` suggests it. The expected wealth at timestep 40 turns out to be: 1.4247570326108. The influence diagram is:



### Exercise 7.14

In order to understand this problem and its utilities and probabilities, we need to look at it from the "player's" perspective. Once he knows that he has a 100 (or  $C$  in general) pound cheque, then there are two possibilities (assuming each has equal probability of 0.5). Either the other cheque is 50 ( $C/2$ ) pounds or 200 ( $2C$ ) pounds. Therefore, we have the following expected utilities (where  $U(\text{high}, \text{change})$  means that the  $C$  or 100 pound cheque is the one with twice the amount of the other cheque, which would be 50 pounds):

$$U(\text{change}) = p(\text{high})U(\text{high}, \text{change}) + p(\text{low})U(\text{low}, \text{change}) = 0.5 \frac{C}{2} + 0.5 * 2C = 1.25 * C$$

$$U(\text{no change}) = p(\text{high})U(\text{high}, \text{no change}) + p(\text{low})U(\text{low}, \text{no change}) = 0.5 * C + 0.5 * C = C$$

Therefore, it is better to change. The same applies if we know the value and it is 100, we simply replace  $C = 100$ . The confusion regarding this problem is that in reality the "experiment"

would be set to either 50-100 or 100-200. In that case (say 50-100), if we had someone change their choice all the time, then it would make no difference. Therefore, if the "player" knew that the case was that it is 50-100, then he would never change if he knew that he got the 100 cheque and if he didn't look at the cheque, then the expected utility would be the same regardless of whether he decides to change or not. The counterintuitive thing about the solution is that, in a sense, it assumes that the experiment can be reproduced many times (as in, every time the player picks a cheque with C or 100 pounds and does not know if the other cheque is  $C/2$  or  $2C$  (50 or 200)), and after running it many times, the player who always changes will have more money than the one who never changes.

## Exercise 9.1

For the first part of the exercise, we implemented the simple counting algorithm for the maximum likelihood approach. For the second part our MATLAB code prints:

The probability that there is a fuse assembly malfunction with ML: 1.000000

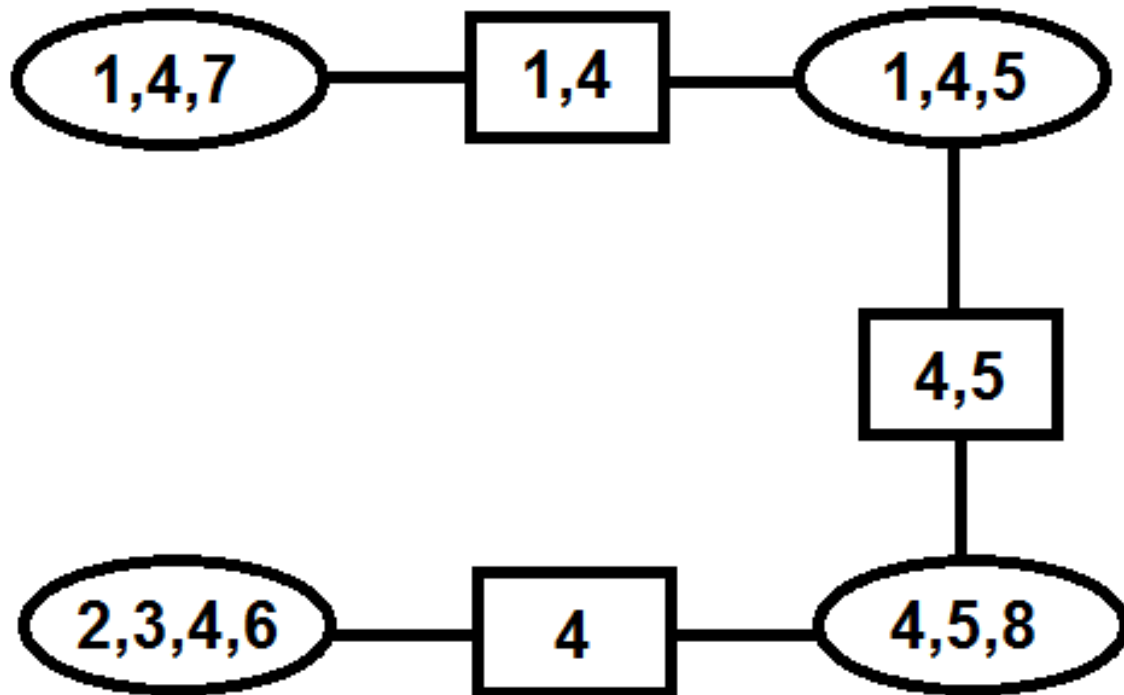
For the third part, according to equations 9.4.16 and 9.4.19 and for a flat Beta prior, we simply need to add a 1 to the numerators and a 2 to the denominators during the calculation. Since we have very little data, this in effect reduces the polarization of the probabilities and gives a more conservative probability. MATLAB code prints:

The probability that there is a fuse assembly malfunction with Bayesian Learning: 0.618615

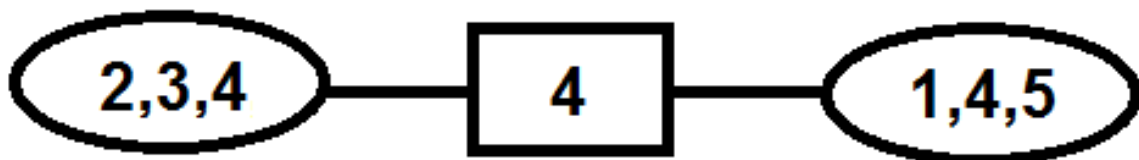
For the 4th part after running the code, we manually inspected the final tables and found out that the most likely state is that there is a fuse assembly malfunction and no other problems, just as we programmed MATLAB to print:

After inspection of the tables of `jtpotfullabsorbcond` and `jtsepfullabsorbcond`, the most likely state is that there is a fuse assembly malfunction and no other problems

For the 5th part, after running the `squeezepots` and `jtree` functions from BRML, the junction tree returned is the following (1: Fuse, 2: Drum, 3: Toner, 4: Paper, 5: Roller, 6: Quality, 7: Wrinkled, 8: Multiple Pages):



Normally we could start the absorption from the 1,4,7 to 1,4,5 and continue. However, we proved in the last assignment, in exercise 6.12, that if a variable exists in only a single clique, then all the potentials that include that variable will exist in that specific clique node. Therefore, whenever we perform the absorption, and we sum over that variable, then it, in effect, gives a new junction tree, which is the marginal of the rest of the variables. Therefore, we can sum over variable 7. This in effect, eliminates the node 1,4,7 and separator 1,4 and we will then need to "move" the whatever is left (the potentials that we assigned) in node 1,4,7 to node 1,4,5. We can simply think of it as a different assignment of the potentials. We can do the same with the variables 6 and 8. For variable 8, it might not be obvious, since it is in the middle of two clique nodes, but the fact remains that by summing the clique potentials in clique 4,5,8 over variable 8, will result in a junction tree with the marginal of the rest of the variables. In the end, after summing over variables 6,7 and 8 and reassigning the potentials, we are left with the junction tree:



We can now perform the max-absorption algorithm. Once again, after inspecting the final tables, we find out that the most likely state is that there is a fuse malfunction, but also poor paper quality and no other problems:

After inspection of the tables of `jtspotfullabsorbcond2` and `jtsepfullabsorbcond2`, the most likely state is that there is a fuse assembly malfunction and poor paper quality and no other problems

MATLAB code:

```
1 clear all; clc; close all;
2 load('printer.mat');
3 import brml.*

5 Fuse = 1;
6 Drum = 2;
7 Toner = 3;
8 Paper = 4;
9 Roller = 5;
10 Burning = 6;
11 Quality = 7;
12 Wrinkled = 8;
13 MultiplePages = 9;
14 PaperJam = 10;

15 fa=1;
16 tr=2;

17 %Part 1
18 tempopot = array(Fuse);
19 tempopot.table(fa) = sum(x(Fuse,:) == fa)/length(x(Fuse,:));
20 tempopot.table(tr) = 1 - tempopot.table(fa);
21 pot{Fuse} = tempopot;

22 tempopot = array(Drum);
23 tempopot.table(fa) = sum(x(Drum,:) == fa)/length(x(Drum,:));
24 tempopot.table(tr) = 1 - tempopot.table(fa);
25 pot{Drum} = tempopot;

26 tempopot = array(Toner);
27 tempopot.table(fa) = sum(x(Toner,:) == fa)/length(x(Toner,:));
28 tempopot.table(tr) = 1 - tempopot.table(fa);
29 pot{Toner} = tempopot;

30 tempopot = array(Paper);
31 tempopot.table(fa) = sum(x(Paper,:) == fa)/length(x(Paper,:));
32 tempopot.table(tr) = 1 - tempopot.table(fa);
33 pot{Paper} = tempopot;

34 tempopot = array(Roller);
35 tempopot.table(fa) = sum(x(Roller,:) == fa)/length(x(Roller,:));
36 tempopot.table(tr) = 1 - tempopot.table(fa);
37 pot{Roller} = tempopot;

38 tempopot = array([Burning Fuse]);
39 tempopot.table(fa,fa) = sum(x(Burning,x(Fuse,:) == fa)==fa)/length(find(x(Fuse,:)
    == fa));
40 tempopot.table(fa,tr) = sum(x(Burning,x(Fuse,:) == tr)==fa)/length(find(x(Fuse,:)
    == tr));
41 tempopot.table(tr,:) = 1 - tempopot.table(fa,:);
42 pot{Burning} = condpot(tempopot,Burning,Fuse);

43 tempopot = array([Quality Drum Toner Paper]);
44 tempopot.table(fa,fa,fa,fa) = sum(x(Quality,x(Drum,:) == fa & x(Toner,:) == fa & x
    (Paper,:) == fa)==fa)/length(find((x(Drum,:) == fa & x(Toner,:) == fa & x(
    Paper,:) == fa)==fa));
45 tempopot.table(fa,fa,fa,tr) = sum(x(Quality,x(Drum,:) == fa & x(Toner,:) == fa & x
    (Paper,:) == tr)==fa)/length(find((x(Drum,:) == fa & x(Toner,:) == fa & x(
    Paper,:) == tr)==fa));
46 tempopot.table(fa,fa,tr,fa) = sum(x(Quality,x(Drum,:) == fa & x(Toner,:) == tr & x
    (Paper,:) == fa)==fa)/length(find((x(Drum,:) == fa & x(Toner,:) == tr & x(
    Paper,:) == fa)==fa));
```

```

55 tempptot.table(fa,fa,tr,tr) = sum(x(Quality,x(Drum,:) == fa & x(Toner,:) == tr & x(
    (Paper,:) == tr)==fa)/length(find((x(Drum,:) == fa & x(Toner,:) == tr & x(
    Paper,:) == tr)==fa));
tempptot.table(fa,tr,fa,fa) = sum(x(Quality,x(Drum,:) == tr & x(Toner,:) == fa & x(
    (Paper,:) == fa)==fa)/length(find((x(Drum,:) == tr & x(Toner,:) == fa & x(
    Paper,:) == fa)==fa));
57 tempptot.table(fa,tr,fa,tr) = sum(x(Quality,x(Drum,:) == tr & x(Toner,:) == fa & x(
    (Paper,:) == tr)==fa)/length(find((x(Drum,:) == tr & x(Toner,:) == fa & x(
    Paper,:) == tr)==fa));
tempptot.table(fa,tr,tr,fa) = sum(x(Quality,x(Drum,:) == tr & x(Toner,:) == tr & x(
    (Paper,:) == fa)==fa)/length(find((x(Drum,:) == tr & x(Toner,:) == tr & x(
    Paper,:) == fa)==fa));
59 tempptot.table(fa,tr,tr,tr) = sum(x(Quality,x(Drum,:) == tr & x(Toner,:) == tr & x(
    (Paper,:) == tr)==fa)/length(find((x(Drum,:) == tr & x(Toner,:) == tr & x(
    Paper,:) == tr)==fa));
tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:);
61 pot{Quality} = condpot(tempptot,Quality,[Drum Toner Paper]);

63 tempptot = array([Wrinkled Fuse Paper]);
tempptot.table(fa,fa,fa) = sum(x(Wrinkled,x(Fuse,:) == fa & x(Paper,:) == fa)==fa)
    /length(find((x(Fuse,:) == fa & x(Paper,:) == fa)==fa));
65 tempptot.table(fa,fa,tr) = sum(x(Wrinkled,x(Fuse,:) == fa & x(Paper,:) == tr)==fa)
    /length(find((x(Fuse,:) == fa & x(Paper,:) == tr)==fa));
tempptot.table(fa,tr,fa) = sum(x(Wrinkled,x(Fuse,:) == tr & x(Paper,:) == fa)==fa)
    /length(find((x(Fuse,:) == tr & x(Paper,:) == fa)==fa));
67 tempptot.table(fa,tr,tr) = sum(x(Wrinkled,x(Fuse,:) == tr & x(Paper,:) == tr)==fa)
    /length(find((x(Fuse,:) == tr & x(Paper,:) == tr)==fa));
tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:);
69 pot{Wrinkled} = condpot(tempptot,Wrinkled,[Fuse Paper]);

71 tempptot = array([MultiplePages Paper Roller]);
tempptot.table(fa,fa,fa) = sum(x(MultiplePages,x(Paper,:) == fa & x(Roller,:) ==
    fa)==fa)/length(find((x(Paper,:) == fa & x(Roller,:) == fa)==fa));
73 tempptot.table(fa,fa,tr) = sum(x(MultiplePages,x(Paper,:) == fa & x(Roller,:) ==
    tr)==fa)/length(find((x(Paper,:) == fa & x(Roller,:) == tr)==fa));
tempptot.table(fa,tr,fa) = sum(x(MultiplePages,x(Paper,:) == tr & x(Roller,:) ==
    fa)==fa)/length(find((x(Paper,:) == tr & x(Roller,:) == fa)==fa));
75 tempptot.table(fa,tr,tr) = sum(x(MultiplePages,x(Paper,:) == tr & x(Roller,:) ==
    tr)==fa)/length(find((x(Paper,:) == tr & x(Roller,:) == tr)==fa));
tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:);
77 pot{MultiplePages} = condpot(tempptot,MultiplePages,[Paper Roller]);

79 tempptot = array([PaperJam Fuse Roller]);
tempptot.table(fa,fa,fa) = sum(x(PaperJam,x(Fuse,:) == fa & x(Roller,:) == fa)==fa)
    /length(find((x(Fuse,:) == fa & x(Roller,:) == fa)==fa));
81 tempptot.table(fa,fa,tr) = sum(x(PaperJam,x(Fuse,:) == fa & x(Roller,:) == tr)==fa)
    /length(find((x(Fuse,:) == fa & x(Roller,:) == tr)==fa));
tempptot.table(fa,tr,fa) = sum(x(PaperJam,x(Fuse,:) == tr & x(Roller,:) == fa)==fa)
    /length(find((x(Fuse,:) == tr & x(Roller,:) == fa)==fa));
83 tempptot.table(fa,tr,tr) = sum(x(PaperJam,x(Fuse,:) == tr & x(Roller,:) == tr)==fa)
    /length(find((x(Fuse,:) == tr & x(Roller,:) == tr)==fa));
tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:);
85 pot{PaperJam} = condpot(tempptot,PaperJam,[Fuse Roller]);

87 %Part 2
Jointpot = multpots(pot);
89 FuseML = setpot(sumpot(Jointpot,[Drum Toner Paper Roller]),[Burning PaperJam
    Quality Wrinkled MultiplePages],[tr tr fa fa fa]);
FuseML.table = FuseML.table/sum(FuseML.table);
91 fprintf('The probability that there is a fuse assembly malfunction with ML: %f\n',
    FuseML.table(tr));

93 %Part 3

```

```

tempptot = array(Fuse);
95 tempptot.table(fa) = (1+sum(x(Fuse,:) == fa))/(2+length(x(Fuse,:)));
tempptot.table(tr) = 1 - tempptot.table(fa);
97 Bayespot{Fuse} = tempptot;

99 tempptot = array(Drum);
tempptot.table(fa) = (1+sum(x(Drum,:) == fa))/(2+length(x(Drum,:)));
101 tempptot.table(tr) = 1 - tempptot.table(fa);
Bayespot{Drum} = tempptot;

103 tempptot = array(Toner);
105 tempptot.table(fa) = (1+sum(x(Toner,:) == fa))/(2+length(x(Toner,:)));
tempptot.table(tr) = 1 - tempptot.table(fa);
107 Bayespot{Toner} = tempptot;

109 tempptot = array(Paper);
tempptot.table(fa) = (1+sum(x(Paper,:) == fa))/(2+length(x(Paper,:)));
111 tempptot.table(tr) = 1 - tempptot.table(fa);
Bayespot{Paper} = tempptot;

113 tempptot = array(Roller);
115 tempptot.table(fa) = (1+sum(x(Roller,:) == fa))/(2+length(x(Roller,:)));
tempptot.table(tr) = 1 - tempptot.table(fa);
117 Bayespot{Roller} = tempptot;

119 tempptot = array([Burning Fuse]);
tempptot.table(fa, fa) = (1+sum(x(Burning, x(Fuse,:) == fa)==fa))/(2+length(find(x(
    Fuse,:) == fa)));
121 tempptot.table(fa, tr) = (1+sum(x(Burning, x(Fuse,:) == tr)==fa))/(2+length(find(x(
    Fuse,:) == tr)));
tempptot.table(tr, :) = 1 - tempptot.table(fa, :);
123 Bayespot{Burning} = condpot(tempptot, Burning, Fuse);

125 tempptot = array([Quality Drum Toner Paper]);
tempptot.table(fa, fa, fa, fa) = (1+sum(x(Quality, x(Drum,:) == fa & x(Toner,:) == fa
    & x(Paper,:) == fa)==fa))/(2+length(find((x(Drum,:) == fa & x(Toner,:) == fa &
    x(Paper,:) == fa)==fa)));
127 tempptot.table(fa, fa, fa, tr) = (1+sum(x(Quality, x(Drum,:) == fa & x(Toner,:) == fa
    & x(Paper,:) == tr)==fa))/(2+length(find((x(Drum,:) == fa & x(Toner,:) == fa &
    x(Paper,:) == tr)==fa)));
tempptot.table(fa, fa, tr, fa) = (1+sum(x(Quality, x(Drum,:) == fa & x(Toner,:) == tr
    & x(Paper,:) == fa)==fa))/(2+length(find((x(Drum,:) == fa & x(Toner,:) == tr &
    x(Paper,:) == fa)==fa)));
129 tempptot.table(fa, fa, tr, tr) = (1+sum(x(Quality, x(Drum,:) == fa & x(Toner,:) == tr
    & x(Paper,:) == tr)==fa))/(2+length(find((x(Drum,:) == fa & x(Toner,:) == tr &
    x(Paper,:) == tr)==fa)));
tempptot.table(fa, tr, fa, fa) = (1+sum(x(Quality, x(Drum,:) == tr & x(Toner,:) == fa
    & x(Paper,:) == fa)==fa))/(2+length(find((x(Drum,:) == tr & x(Toner,:) == fa &
    x(Paper,:) == fa)==fa)));
131 tempptot.table(fa, tr, fa, tr) = (1+sum(x(Quality, x(Drum,:) == tr & x(Toner,:) == fa
    & x(Paper,:) == tr)==fa))/(2+length(find((x(Drum,:) == tr & x(Toner,:) == fa &
    x(Paper,:) == tr)==fa)));
tempptot.table(fa, tr, tr, fa) = (1+sum(x(Quality, x(Drum,:) == tr & x(Toner,:) == tr
    & x(Paper,:) == fa)==fa))/(2+length(find((x(Drum,:) == tr & x(Toner,:) == tr &
    x(Paper,:) == fa)==fa)));
133 tempptot.table(fa, tr, tr, tr) = (1+sum(x(Quality, x(Drum,:) == tr & x(Toner,:) == tr
    & x(Paper,:) == tr)==fa))/(2+length(find((x(Drum,:) == tr & x(Toner,:) == tr &
    x(Paper,:) == tr)==fa)));
tempptot.table(tr, :, :, :) = 1 - tempptot.table(fa, :, :, :);
135 Bayespot{Quality} = condpot(tempptot, Quality, [Drum Toner Paper]);

137 tempptot = array([Wrinkled Fuse Paper]);
tempptot.table(fa, fa, fa) = (1+sum(x(Wrinkled, x(Fuse,:) == fa & x(Paper,:) == fa)==

```

```

    fa))/(2+length(find((x(Fuse,:) == fa & x(Paper,:) == fa)==fa)));
139 tempptot.table(fa,fa,tr) = (1+sum(x(Wrinkled,x(Fuse,:) == fa & x(Paper,:) == tr)==
    fa))/(2+length(find((x(Fuse,:) == fa & x(Paper,:) == tr)==fa)));
    tempptot.table(fa,tr,fa) = (1+sum(x(Wrinkled,x(Fuse,:) == tr & x(Paper,:) == fa)==
    fa))/(2+length(find((x(Fuse,:) == tr & x(Paper,:) == fa)==fa)));
141 tempptot.table(fa,tr,tr) = (1+sum(x(Wrinkled,x(Fuse,:) == tr & x(Paper,:) == tr)==
    fa))/(2+length(find((x(Fuse,:) == tr & x(Paper,:) == tr)==fa)));
    tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:) ;
143 Bayespot{Wrinkled} = condpot(tempptot,Wrinkled,[Fuse Paper]);

145 tempptot = array([MultiplePages Paper Roller]);
    tempptot.table(fa,fa,fa) = (1+sum(x(MultiplePages,x(Paper,:) == fa & x(Roller,:)
    == fa)==fa))/(2+length(find((x(Paper,:) == fa & x(Roller,:) == fa)==fa)));
147 tempptot.table(fa,fa,tr) = (1+sum(x(MultiplePages,x(Paper,:) == fa & x(Roller,:)
    == tr)==fa))/(2+length(find((x(Paper,:) == fa & x(Roller,:) == tr)==fa)));
    tempptot.table(fa,tr,fa) = (1+sum(x(MultiplePages,x(Paper,:) == tr & x(Roller,:)
    == fa)==fa))/(2+length(find((x(Paper,:) == tr & x(Roller,:) == fa)==fa)));
149 tempptot.table(fa,tr,tr) = (1+sum(x(MultiplePages,x(Paper,:) == tr & x(Roller,:)
    == tr)==fa))/(2+length(find((x(Paper,:) == tr & x(Roller,:) == tr)==fa)));
    tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:) ;
151 Bayespot{MultiplePages} = condpot(tempptot,MultiplePages,[Paper Roller]);

153 tempptot = array([PaperJam Fuse Roller]);
    tempptot.table(fa,fa,fa) = (1+sum(x(PaperJam,x(Fuse,:) == fa & x(Roller,:) == fa)
    ==fa))/(2+length(find((x(Fuse,:) == fa & x(Roller,:) == fa)==fa)));
155 tempptot.table(fa,fa,tr) = (1+sum(x(PaperJam,x(Fuse,:) == fa & x(Roller,:) == tr)
    ==fa))/(2+length(find((x(Fuse,:) == fa & x(Roller,:) == tr)==fa)));
    tempptot.table(fa,tr,fa) = (1+sum(x(PaperJam,x(Fuse,:) == tr & x(Roller,:) == fa)
    ==fa))/(2+length(find((x(Fuse,:) == tr & x(Roller,:) == fa)==fa)));
157 tempptot.table(fa,tr,tr) = (1+sum(x(PaperJam,x(Fuse,:) == tr & x(Roller,:) == tr)
    ==fa))/(2+length(find((x(Fuse,:) == tr & x(Roller,:) == tr)==fa)));
    tempptot.table(tr,,:,:) = 1 - tempptot.table(fa,,:,:) ;
159 Bayespot{PaperJam} = condpot(tempptot,PaperJam,[Fuse Roller]);

161 BayesJointpot = multpots(Bayespot);
    FuseBayes = setpot(sumpot(BayesJointpot,[Drum Toner Paper Roller]),[Burning
    PaperJam Quality Wrinkled MultiplePages],[tr tr fa fa fa]);
163 FuseBayes.table = FuseBayes.table/sum(FuseBayes.table);
    fprintf('The probability that there is a fuse assembly malfunction with Bayesian
    Learning: %f\n',FuseBayes.table(tr));
165
166 %part 4
167 for i = 1:1:10
    potcond{i} = setpot(Bayespot{i},[Burning PaperJam Quality Wrinkled
    MultiplePages],[tr tr fa fa fa]);
169 end
    [newpot, ~, ~, ~] = squeezeopts(potcond);
171 [jtpotcond, jtsepcond, infostructcond]=jtree(newpot);
    [jtpotfullabsorbcond, jtsepfullabsorbcond, Zcond]=absorption(jtpotcond,jtsepcond,
    infostructcond,'max');
173 fprintf('After inspection of the tables of jtpotfullabsorbcond and
    jtsepfullabsorbcond, the most likely state is that there is a fuse assembly
    malfunction and no other problems\n');

175 %part 5
    for i = 1:1:10
177 potcond2{i} = setpot(Bayespot{i},[Burning PaperJam],[tr tr]);
    end
179 [newpot2, ~, ~, ~] = squeezeopts(potcond2);
    [jtpotcond2, jtsepcond2, infostructcond2]=jtree(newpot2);
181 for i = 1:1:length(jtpotcond2)
    jtpotcond2{i} = sumpot(jtpotcond2{i},[6 7 8]);
183 end

```

```

185 jtpotcond2{1} = multpots([jtpotcond2{1} jtpotcond2{2} jtpotcond2{3}]);
jtpotcond2{2} = jtpotcond2{4};
jtpotcond2 = jtpotcond2([1 2]);
187 jtsepcond2 = jtsepcond2(1);
infostructcond2.cliquevariables{1} = [1 4 5];
189 infostructcond2.cliquevariables{2} = [2 3 4];
infostructcond2.cliquevariables = infostructcond2.cliquevariables([1 2]);
191 infostructcond2.separator = infostructcond2.separator([3]);
infostructcond2.cliquevariables = sparse([0 1; 1 0]);
193 infostructcond2.sepind = sparse([1 2], [2 1], [1 1]);
infostructcond2.EliminationSchedule = [1 2];
195 [jtpotfullabsorbcond2, jtsepfullabsorbcond2, Zcond2]=absorption(jtpotcond2,
jtsepcond2, infostructcond2, 'max');
fprintf('After inspection of the tables of jtpotfullabsorbcond2 and
jtsepfullabsorbcond2, the most likely state is that there is a fuse assembly
malfunction and poor paper quality and no other problems\n');

```

## Exercise 9.9

Starting with equation 9.4.53, we can use the equation 8.3.30 for each of the normalization constants of the Dirichlet distributions:

$$p(D) = \prod_k \prod_j \frac{Z(\mathbf{u}'(v_k; j))}{Z(\mathbf{u}(v_k; j))}$$

$$Z(\mathbf{u}) = \frac{\prod_i \Gamma(u_i)}{\Gamma(\sum_i u_i)}$$

Therefore:

$$Z(\mathbf{u}'(v_k; j)) = \frac{\prod_i \Gamma(u'_i(v_k; j))}{\Gamma(\sum_i u'_i(v_k; j))}$$

$$Z(\mathbf{u}(v_k; j)) = \frac{\prod_i \Gamma(u_i(v_k; j))}{\Gamma(\sum_i u_i(v_k; j))}$$

So, by inputting them into the initial equation we get:

$$p(D) = \prod_k \prod_j \frac{\prod_i \Gamma(u'_i(v_k; j))}{\Gamma(\sum_i u'_i(v_k; j))} \frac{\Gamma(\sum_i u_i(v_k; j))}{\prod_i \Gamma(u_i(v_k; j))} =$$

$$p(D) = \prod_k \prod_j \frac{\Gamma(\sum_i u_i(v_k; j))}{\Gamma(\sum_i u'_i(v_k; j))} \frac{\prod_i \Gamma(u'_i(v_k; j))}{\prod_i \Gamma(u_i(v_k; j))} =$$

$$p(D) = \prod_k \prod_j \frac{\Gamma(\sum_i u_i(v_k; j))}{\Gamma(\sum_i u'_i(v_k; j))} \prod_i \left[ \frac{\Gamma(u'_i(v_k; j))}{\Gamma(u_i(v_k; j))} \right]$$

## Exercise 9.10

Part 1

Given the arbitrary ancestral ordering  $a: x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ , the number of belief networks will be:  $\#N_a = \#a(x_1) * \#a(x_2) * \dots * \#a(x_8)$ , where  $\#a(x_i)$  is the number of different



combinations/possible ancestors of  $x_i$ . As such, we have:

$$\begin{aligned}
\#a(x_1) &= 1 = 1 \\
\#a(x_2) &= 1 + 1 = 2 \\
\#a(x_3) &= 1 + \binom{2}{1} + \binom{2}{2} = 4 \\
\#a(x_4) &= 1 + \binom{3}{1} + \binom{3}{2} = 7 \\
\#a(x_5) &= 1 + \binom{4}{1} + \binom{4}{2} = 11 \\
\#a(x_6) &= 1 + \binom{5}{1} + \binom{5}{2} = 16 \\
\#a(x_7) &= 1 + \binom{6}{1} + \binom{6}{2} = 22 \\
\#a(x_8) &= 1 + \binom{7}{1} + \binom{7}{2} = 29
\end{aligned}$$

Therefore:

$$\#N_a = 1 * 2 * 4 * 7 * 11 * 16 * 22 * 29 = 6288128$$

## Part 2

Due to the decomposability of the BD score, we can optimize it for each variable independently. Therefore the total number of belief networks we will need to examine is:

$$O(N_a) = \#a(x_1) + (\#a(x_2) - 1) + \dots + (\#a(x_8) - 1) = 1 + 1 + 3 + 6 + 10 + 15 + 21 + 29 = 86seconds$$

The fact that we only need to sum instead of multiply is thanks to the decomposability property. The minus 1 to each element can be understood if, for example, we only have 3 variables. In this case we could give a state for what parents each variable has. For example state 111 means that none of the variables have a parent, whereas state 121 means that the second variable has the first variable as a parent. We would thus first examine state 111. Then state 121. By now, we would know the parents of the first and second variable. Let us assume that the second variable also has no parents. We would then proceed to check states 112, 113 and 114. Due to the fact that we don't need to recheck state 111 is the reason for the minus one.

## Part 3

The only thing left is to find all the permutations of ancestral orderings. Therefore the total time will be  $O(N) = 8!O(N_a) = 3467520seconds$

## Exercise 9.13

MATLAB code:

```

function A = ChowLiu(X)
2 import brml.*
  [D,N] = size(X);
4 W = zeros(D);
  %Find all mutual information
6 for i = 1:1:D
    for j = 1:1:D
8         W(i,j) = MIemp(X(i,:),X(j,:),max(X(i,:),),max(X(j,:),));
    end
10 end

```

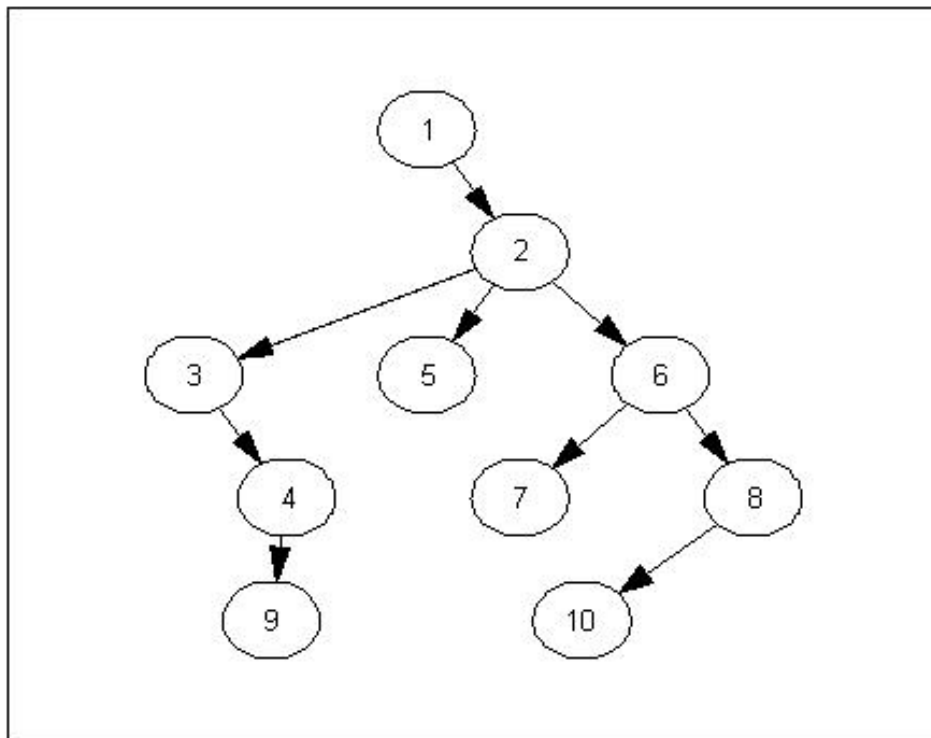
```

12 %Spanning tree
13 [Atree, ~, ~]=spantree(W);
14 %DAG
15 A = Atree;
16 Ordering = 1;
17 for counter = 1:1:D
18     node = Ordering(counter);
19     A(setdiff(1:D, Ordering(1:counter)), node) = 0;
20     Ordering = [Ordering find(A(node,:))];
21 end
22 drawNet(A);
23 end

```

The code to convert the tree to a DAG finds the ancestral ordering of the variables and on every iteration, makes sure that no "younger" nodes point to the current "older" node.

The Chow Liu tree we get is:



### Exercise 10.3

MATLAB code:

```

2 clear all; clc; close all;
3 import brml.*
4 xP = [1 0 1 1 1 0 1 1;
5       0 0 0 1 0 0 1 1;
6       1 0 0 1 1 0 1 0;
7       0 1 0 0 1 1 0 1;
8       0 0 0 1 1 0 1 1;

```

```

8      0 0 0 1 1 0 0 1]';
xS = [1 1 0 0 0 0 0 0;
10     0 0 1 0 0 0 0 0;
      1 1 0 1 0 0 0 0;
12     1 1 0 1 0 0 0 1;
      1 1 0 1 1 0 0 0;
14     0 0 0 1 0 1 0 0;
      1 1 1 1 1 0 1 0]';
16 [pP, pS, mP, mS]=NaiveBayesTrain(xP,xS);
x = [1 0 0 1 1 1 1 0]';
18 px=NaiveBayesTest(x,pP,pS,mP,mS);
fprintf('The probability that x=(1,0,0,1,1,1,1,0) is about politics is: %f.\n',px
(1));

```

The probability that  $x=(1,0,0,1,1,1,1,0)$  is about politics is: 0.830599.

## Exercise 10.5

Part 1

According to equations (10.2.5) and (10.2.7) we have:

$$\begin{aligned}
 p(c=1) &= \frac{\sum_{n=1}^N I[c^n=1]}{N} \\
 p(x_i=1|c=1) &= \frac{\sum_{n=1}^N I[x_i^n=1, c^n=1]}{\sum_{n=1}^N I[x_i^n=1, c^n=1] + I[x_i^n=0, c^n=1]}, i=1, \dots, D \\
 p(x_i=1|c=0) &= \frac{\sum_{n=1}^N I[x_i^n=1, c^n=0]}{\sum_{n=1}^N I[x_i^n=1, c^n=0] + I[x_i^n=0, c^n=0]}, i=1, \dots, D
 \end{aligned}$$

Part 2

As described in the Classification boundary section of the book, given an input  $x^*$ , we can classify it to be in class  $c=1$  if (equation 10.2.10):

$$\begin{aligned}
 p(c=1|x^*) &> p(c=0|x^*) \\
 \sum_{i=1}^D \log p(x_i^*|c=1) + \log p(c=1) &> \sum_{i=1}^D \log p(x_i^*|c=0) + \log p(c=0)
 \end{aligned}$$

Otherwise, we classify it as belonging to class  $c=0$ .

Part 3

If we assume that  $x_4=1$  means that the word 'viagra' is present in the e-mail, then if it never appears in the training data, it means that:

$$p(x_4=1|c=1) = \frac{\sum_{n=1}^N I[x_4^n=1, c^n=1]}{\sum_{n=1}^N I[x_4^n=1, c^n=1] + I[x_4^n=0, c^n=1]} = \frac{0}{0+N} = 0$$

Which means that:

$$p(c=1|x^*) = \frac{p(x^*|c=1)p(c=1)}{p(x^*)} = \prod_{i=1}^D p(x_i=x_i^*|c=1) \frac{p(c=1)}{p(x^*)} = 0$$

The above is equal to 0 since  $x_4^*=1$  and  $p(x_4=x_4^*|c=1) = p(x_4=1|c=1) = 0$ . This means that an e-mail containing the word 'viagra' will never be classified as spam with the current

training performed. By using a Bayesian approach, as is explained in chapter 10.3, we can use Beta distribution priors and change the learning of the tables to, for example with a flat prior, add a 1 to the numerator and a 2 to the denominator, just as in exercise 9.1 part 3. This, essentially has the effect of enforcing the training to happen as if in the training data all words appear and also are absent at least once. One could produce the same result by adding the e-mails of all ones and all zeros:  $[1,1,\dots,1]$  and  $[0,0,\dots,0]$  to both spam and not-spam training data sets.

Therefore, a spammer should in general use words which do not appear, or rarely appear, in typical spam e-mails. They could also slightly change the words, e.g. 'v1agra' instead of 'viagra'. Also, depending on how "online" the training is (as in, it keeps retraining itself with the new data), one could possibly send a lot of spam with words he does not intend to use in his real spam e-mail and lacking the words that he intends to use in his real spam e-mail, which would polarize the classifier more towards not classifying his real spam e-mails as such.