

Problem Set 3 Solutions

MAS 622J/1.126J: Pattern Recognition and Analysis

(Revised Solution) Due Friday, 10 October 2008

[Note: All instructions to plot data or write a program should be carried out using either Python accompanied by the `matplotlib` package or Matlab. Feel free to use either or both, but in order to maintain a reasonable level of consistency and simplicity we ask that you do not use other software tools.]

Problem 1: Discriminability and ROC

Please download the datasets (dataset1.txt, dataset2.txt, dataset3.txt, dataset4.txt) from the course website. Each dataset includes 1-D data samples from two classes w_1 and w_2 . The first column and the second column of each dataset specify 1000 samples from class w_1 and 1000 samples from class w_2 , respectively.

- a. For each dataset, compute the discriminability $d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\sigma_1^2 + \sigma_2^2}}$ where μ_1 and σ_1 are the mean and standard deviation of the distribution of class w_1 , and μ_2 and σ_2 are the mean and standard deviation of the distribution of class w_2 .

(Answer)

Please refer to the Matlab code for (b). Approximately,

For dataset1: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 1$, $\sigma_2 \simeq 1$, $d' \simeq 1.39$

For dataset2: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 4$, $\sigma_2 \simeq 1$, $d' \simeq 0.46$

For dataset3: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 1$, $\sigma_2 \simeq 4$, $d' \simeq 0.45$

For dataset4: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 4$, $\sigma_2 \simeq 4$, $d' \simeq 0.31$

- b. Now we compute the ROC curve for each dataset. Please plot four ROC curves in the same figure. To do this, we approximate $\mathbf{P}_{\mathbf{TP}} = \mathbf{P}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_2)$ by $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_2)/1000$, and $\mathbf{P}_{\mathbf{FP}} = \mathbf{P}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_1)$ by $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_1)/1000$. Here, for $i=1$ or 2 , $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_i)$ is denotes the number of samples in class w_i whose value is greater than \mathbf{x}^* . Note that \mathbf{N} doesn't denote a normal distribution!

(Answer)

Changing the classifier value (\mathbf{x}^*) from $-\infty$ to ∞ , we can plot the ROC curve for each dataset. Note that when $\mathbf{x}^* = -\infty$, $\mathbf{P}_{\mathbf{TP}} = \mathbf{1}$ and $\mathbf{P}_{\mathbf{FP}} = \mathbf{1}$ (corresponding to the uppermost/rightmost position in the ROC curve), and when $\mathbf{x}^* = \infty$, $\mathbf{P}_{\mathbf{TP}} = \mathbf{0}$ and $\mathbf{P}_{\mathbf{FP}} = \mathbf{0}$ (corresponding to the lowermost/leftmost position in the ROC curve).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
%
% (for Problem 1(a),(b))
```

```

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all

num_samples = 1000; % number of test samples from each class

sprintf('dataset1:')
dd = load('dataset1.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'

figure
hold on
plot(Pf_save, Pd_save, 'r')

sprintf('dataset2:')

```

```

dd = load('dataset2.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'

plot(Pf_save, Pd_save, 'b')

sprintf('dataset3:')
dd = load('dataset3.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

```

```

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'

plot(Pf_save, Pd_save, 'g')

sprintf('dataset4:')
dd = load('dataset4.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));

```

```

% approximate probability of false alarm,  $P(x > x_c \mid x \text{ in } w=1)$ 
Pf = nFP / num_samples;

Pd_save = [Pd_save; Pd];
Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

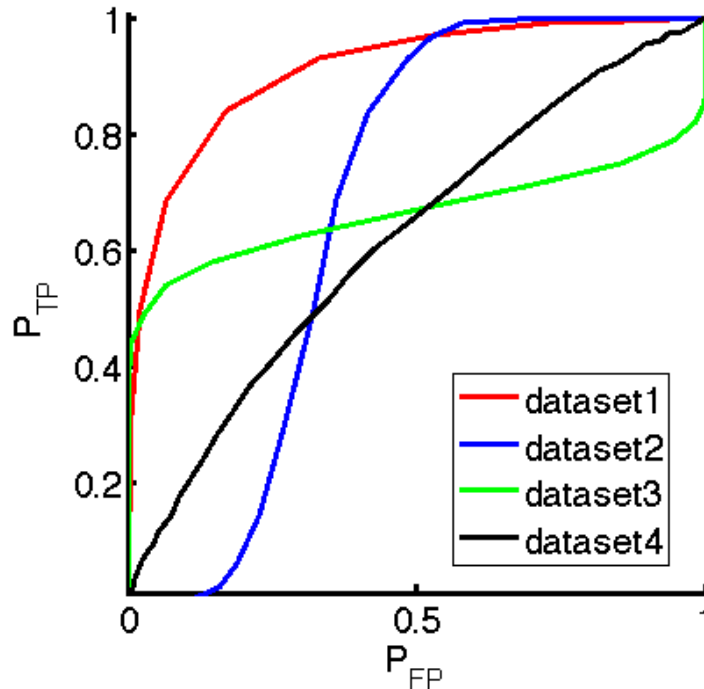
discrim = abs(mean1 - mean2)/sig % d'

plot(Pf_save, Pd_save, 'k')
xlabel('P_{FP}')
ylabel('P_{TP}')
axis equal
axis([0 1 0 1])

legend('dataset1', 'dataset2', 'dataset3', 'dataset4')

hold off

```



- c. For each dataset, plot the two *approximated* probability density functions. Note that the probability density function is the derivative of the cumulative density function. (Do NOT just approximate distributions by Gaussians and draw those approximated Gaussians in this problem.) *Hint:* Use the

same method we use in (b) to get the cumulative distribution. This is called *Monte Carlo* method. Also, note that the derivative (i.e., probability density function $p(x)$) relates to the increment of the cumulative density function $P_X(x)$. That is, $p(x) = \Delta P_X(x)/\Delta x$.

(Answer)

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
%
% (for Problem 1(c))
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function draw_distributions_all()

figure

% dataset1
% two sample distributions
dd = load('dataset1.txt');
draw_distributions(dd, 1);

% dataset2
% two sample distributions
dd = load('dataset2.txt');
draw_distributions(dd, 2);

% dataset3
% two sample distributions
dd = load('dataset3.txt');
draw_distributions(dd, 3);

% dataset4
% two sample distributions
dd = load('dataset4.txt');
draw_distributions(dd, 4);

function draw_distributions(dd, dataset_id)

numr = length(dd);% number of total samples for each distribution
delx = 0.5;

bins = -5:delx:30;
x1 = dd(:,1);
x2 = dd(:,2);

ref = (mu(1)+mu(2))/2;

```

```

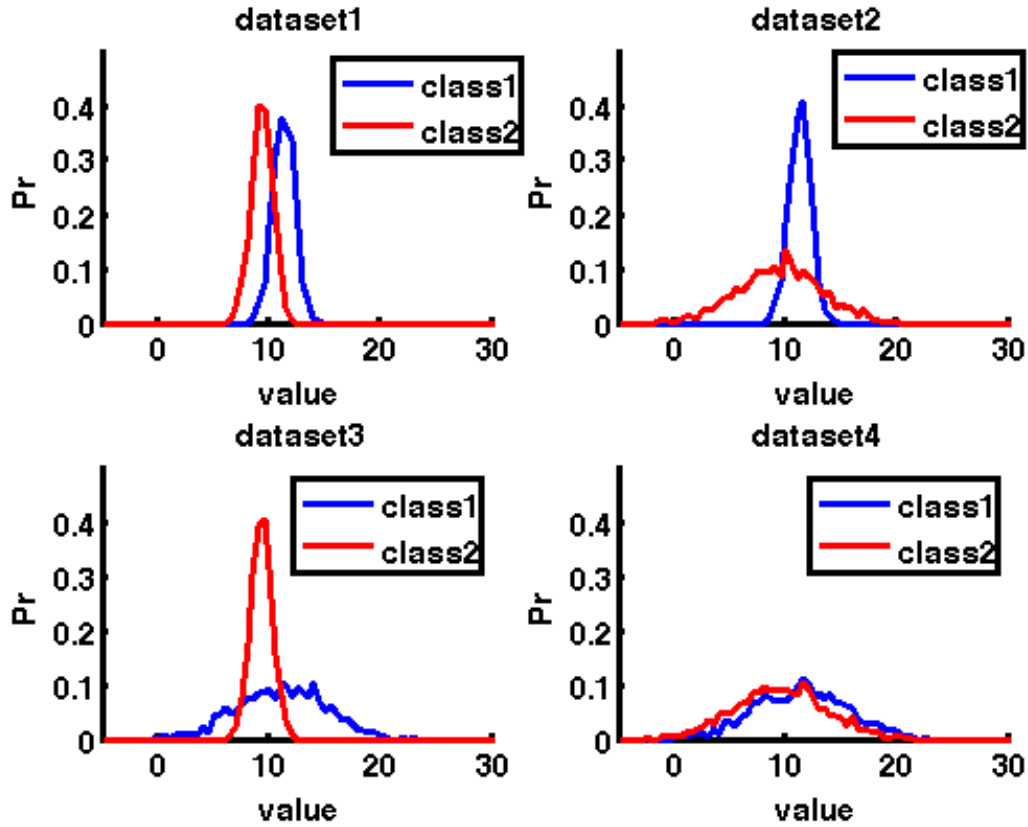
c1 = histc(x1,bins);
c2 = histc(x2,bins);

p1 = c1/delx/numr;
p2 = c2/delx/numr;

% The above is exactly the same as this:
% cc1 = cumsum(c1)/numr;
% for t=1:length(bins)-1
%     p1(t) = (cc1(t+1)-cc1(t))/delx;
% end
% p1(t+1) = 0;
% cc2 = cumsum(c2)/numr;
% for t=1:length(bins)-1
%     p2(t) = (cc2(t+1)-cc2(t))/delx;
% end
% p2(t+1) = 0;

subplot(2,2,dataset_id)
hold on
plot(bins, p2, 'b');
plot(bins, p1, 'r')
axis([-5 30 0 0.5])
legend('class1', 'class2')
xlabel('value')
ylabel('Pr')
titlestr = sprintf('dataset%d', dataset_id);
title(titlestr)
hold off

```



d. How does the discriminability relate to the ROC curve?

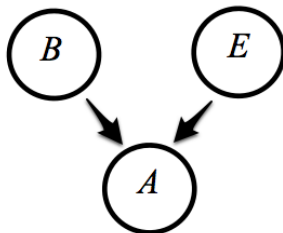
(Answer)

First, comparing dataset1 ($d' = 1.39$) with dataset4 ($d' = 0.31$), we can see that the two distributions in dataset1 are more discriminable than the ones in dataset4 (See the distribution plot in (c)). Also, from the ROC plot in (b), we find that better performance (here, meaning higher $\mathbf{P}_{\mathbf{TP}}$ and lower $\mathbf{P}_{\mathbf{FP}}$ with the optimal classifier location) is achieved on dataset1, compared with dataset4. In general, for two gaussian distributions, as d' becomes bigger, $\mathbf{P}_{\mathbf{TP}} - \mathbf{P}_{\mathbf{FP}}$ becomes bigger (Refer to the problem 5 (a) in our Problem Set 2).

Second, comparing dataset2 ($d' = 0.46$) with dataset3 ($d' = 0.45$), we can see that the discriminability of the two distributions in dataset2 is almost the same as that of the ones in dataset3 (See the distribution plot in (c)). Also, from the ROC plot in (b), we find that ROC curves of dataset3 and dataset4 have very different characteristics although the two have similar d' values. Note that depending on the shapes of two distributions (in dataset2, the distribution with higher mean has lower variance, but in dataset3, the distribution with higher mean has higher variance), the characteristic of the corresponding ROC curve is determined. Note that for each dataset, when the change of the classifier value (\mathbf{x}^*) happens at around the mean of the distribution with lower variance, the ROC curve ($\mathbf{P}_{\mathbf{TP}}$ for dataset2, $\mathbf{P}_{\mathbf{FP}}$ for dataset3) radically changes.

Problem 2: Conditional Dependence

Consider the following Bayesian Network where the three events are B (Burglary), E (Earthquake) and A (Alarm). Assume that the three nodes are binary nodes that can take on the value f (false) or t (true).



Here, B and E are marginally independent. That is, $P(B, E) = P(B)P(E)$ (Note that this relation implies $P(B|E) = P(B)$ or $P(E|B) = P(E)$.) Now we want to show B and E are conditionally dependent given A . In other words, $P(B, E|A) \neq P(B|A)P(E|A)$.

- a. Prove that the relation $P(B, E|A) \neq P(B|A)P(E|A)$ implies $P(B|E, A) \neq P(B|A)$. Note that by the symmetry between B and E , this also means $P(E|B, A) \neq P(E|A)$.

(Answer) From $P(B, E|A) \neq P(B|A)P(E|A)$, we get the relation $\frac{P(B, E|A)}{P(E|A)} \neq P(B|A)$

Also, we know that $P(B|E, A) = \frac{P(B, E|A)}{P(E|A)}$ by Bayes' rule.

Therefore, $P(B|E, A) \neq P(B|A)$

- b. First, we consider a very simple case. Suppose $A = B + E$ where the $+$ sign means "Logical OR." This means that B and E are independent deterministic causes of A . Construct the CPT (conditional probability table) for $P(A | B, E)$.

B, E	$P(A = f B, E), P(A = t B, E)$
f, f	1 0
t, f	0 1
f, t	0 1
t, t	0 1

- c. For the obtained CPT above, prove that $P(B = t|A = t) = P(B = t)/P(A = t)$. Which is greater between $P(B = t|A = t)$ and $P(B = t)$? What is the meaning of this?

(Answer) $P(B = t|A = t) = P(A = t, B = t)/P(A = t)$ by Bayes' rule.

Now for $P(A = t, B = t)$,

$$\begin{aligned}
 & P(A = t, B = t) \\
 &= \sum_{E=t, f} P(A = t, B = t, E) \quad (\text{by marginalization}) \\
 &= \sum_{E=t, f} P(B = t)P(E)P(A = t|B = t, E) \quad (\text{from the graphical model}) \\
 &= P(B = t)P(E = t)P(A = t|B = t, E = t) + P(B = t)P(E = f)P(A = t|B = t, E = f) \\
 &= P(B = t)P(E = t) + P(B = t)P(E = f) \\
 &= P(B = t)(P(E = t) + P(E = f)) \\
 &= P(B = t)
 \end{aligned}$$

Therefore, $P(B = t|A = t) = P(B = t)/P(A = t)$

Since $P(A = t) \leq 1$, $P(B = t|A = t) \geq P(B = t)$

Note that $P(B = t)$ means your belief on $B = t$ (Burglary happened) without being given any information about E and A . However, $P(B = t|A = t)$ means your belief on $B = t$ (Burglary happened) when you know $A = t$ (Alarm happened). Thus, $P(B = t|A = t) \geq P(B = t)$ means that your belief of $B = t$ (Burglary happend) increases when you know $A = t$ (Alarm happened).

- d. For the obtained CPT above, prove that $P(A = t) = P(B = t) + P(E = t) - P(B = t)P(E = t)$. If $P(B = t)$ and $P(E = t)$ are small, what happens to $P(B = t|A = t)$, compared with $P(B = t)$?

(Answer)

$$\begin{aligned}
 P(A = t) &= \sum_{B=t,f} \sum_{E=t,f} P(A = t, B, E) \\
 &= \sum_{B=t,f} \sum_{E=t,f} P(B)P(E)P(A = t|B, E) \\
 &= P(B = t)P(E = t)P(A = t|B = t, E = t) + P(B = t)P(E = f)P(A = t|B = t, E = f) \\
 &\quad + P(B = f)P(E = t)P(A = t|B = f, E = t) + P(B = f)P(E = f)P(A = t|B = f, E = f) \\
 &= P(B = t)P(E = t) + P(B = t)P(E = f) + P(B = f)P(E = t) \\
 &= P(B = t)(P(E = t) + P(E = f)) + (1 - P(B = t))P(E = t) \\
 &= P(B = t) + P(E = t) - P(B = t)P(E = t)
 \end{aligned}$$

Thus, as $P(B = t)$ and $P(E = t)$ gets smaller, $P(A = t)$ also gets smaller. Since $P(B = t|A = t) = P(B = t)/P(A = t)$ (from (d)), $P(B = t|A = t)$ becomes much greater than $P(B = t)$.

- e. For the obtained CPT above, prove that $P(B = t|E = t, A = t) = P(B = t)$. Which is greater between $P(B = t|E = t, A = t)$ and $P(B = t|A = t)$? What is the meaning of this? *Hint*: Note that the observation of Earthquake ($E = t$) alone is enough to explain the cause of Alarm ($A = t$). This occurrence is called “Explaining Away.”

(Answer)

$$\begin{aligned}
 P(B = t|E = t, A = t) &= \frac{P(B=t, E=t, A=t)}{P(E=t, A=t)} \\
 &= \frac{P(B=t)P(E=t)P(A=t|B=t, E=t)}{P(E=t, A=t)} \quad (\text{from the graphical model}) \\
 &= \frac{P(B=t)P(E=t)P(A=t|B=t, E=t)}{P(E=t)} \quad (\text{since } P(E = t, A = t) = P(E = t)) \\
 &= P(B = t)
 \end{aligned}$$

Note that in (c), we proved $P(B = t, A = t) = P(B = t)$. In like manner, we can prove $P(E = t, A = t) = P(E = t)$.

Since $P(B = t|E = t, A = t) = P(B = t)$ and $P(B = t|A = t) \geq P(B = t)$ (from (c)), we get $P(B = t|A = t) \geq P(B = t) = P(B = t|E = t, A = t)$

Note that $P(B = t|E = t, A = t)$ denotes your belief on $B = t$ (Burglary happened) when you know both Earthquake and Alarm happened ($E = t, A = t$). Here the observation of Earthquake ($E = t$) alone is enough to explain the cause of Alarm ($A = t$). Therefore, since the fact that Earthquake happened can “explain away” the fact that Alarm happened, there is no increase in your belief on $B = t$ (Burglary happened). Thus, $P(B = t|E = t, A = t) = P(B = t)$.

- f. We assume that $P(B = t) = 0.1$ and $P(E = t) = 0.01$. Calculate $P(A = t)$, $P(B = t|A = t)$, $P(B = t|E = t, A = t)$, $P(E = t|A = t)$, $P(E = t|B = t, A = t)$ by hand.

(Answer)

$$\begin{aligned}
 P(A = t) &= P(B = t) + P(E = t) - P(B = t)P(E = t) = 0.1 + 0.01 - 0.1 \times 0.01 = 0.109 \\
 P(B = t|A = t) &= P(B = t)/P(A = t) = 0.1/0.109 = 0.9174 \\
 P(B = t|E = t, A = t) &= P(B = t) = 0.1
 \end{aligned}$$

$$P(E = t|A = t) = P(E = t)/P(A = t) = 0.09174$$

$$P(E = t|B = t, A = t) = P(E = t) = 0.01$$

- g. Now install Kevin Murphy's Bayes Net Toolbox for Matlab. You can download the toolbox and find the install and usage instructions here:

<http://bnt.sourceforge.net/>

We compute the above $P(A = t)$, $P(B = t|A = t)$, $P(B = t|E = t, A = t)$, $P(E = t|A = t)$, $P(E = t|B = t, A = t)$ using this toolbox. To help you use the toolbox, we provide you with the basic Matlab code for this problem.

Download the Matlab code “**conditional_dep.m**” from the course website. Fill in the core part in the Matlab code and compute $P(A = t)$, $P(B = t|A = t)$, $P(B = t|E = t, A = t)$, $P(E = t|A = t)$, $P(E = t|B = t, A = t)$. Are the computed values the same to what you've obtained by hand?

(Answer)

```
% MIT MAS622j/1.126J, Problem Set 3
% Conditional Dependence, Explaining Away
% (Burglary-->Alarm<---Earthquake example)
%
% (for Problem 2(g))

clear all
N = 3;
dag = zeros(N,N);
B = 1; E = 2; A = 3;
dag(B,A) = 1;
dag(E,A) = 1;

false= 1; true= 2;
ns = 2*ones(1,N); % binary nodes
names = {'B', 'E', 'A'};

% Set up the topology of the Bayes Net
bnet = mk_bnet(dag, ns, 'names', names, 'discrete', 1:N);

% Display the graph: download graph_draw from the link on the class webpage
graph_draw(bnet.dag, 'node_labels', names);

% Set up conditional probability tables for each node
% ### Your code comes here! ###
% Hint: Fill in the row vectors with CPD values
bnet.CPD{B} = tabular_CPD(bnet, B, [0.9 0.1]);
bnet.CPD{E} = tabular_CPD(bnet, E, [0.99 0.01]);
bnet.CPD{A} = tabular_CPD(bnet, A, [1 0 0 0 0 1 1 1]);

% Choose an inference engine
engine = jtree_inf_engine(bnet);
```

```

% First, compute P(A)
evidence = cell(1,N);
[engine, ll] = enter_evidence(engine, evidence);

m = marginal_nodes(engine, A);
disp(sprintf('p(A) = %g', m.T(true)));

% Second, compute P(B|A)
% ### Your code comes here! ###
% Hint: Enter the observation that Alarm = true
evidence = cell(1,N);
evidence{A} = true;
[engine, ll] = enter_evidence(engine, evidence);

m = marginal_nodes(engine, B);
disp(sprintf('p(B|A) = %g', m.T(true)));

% Third, compute P(B|E,A)
% ### Your code comes here! ###
% Hint: Enter the observation that Earthquake = true and Alarm = true
evidence = cell(1,N);
evidence{E} = true;
evidence{A} = true;
[engine, ll] = enter_evidence(engine, evidence);

m = marginal_nodes(engine, B);
disp(sprintf('p(B|E,A) = %g', m.T(true)));

% Fourth, compute P(E|A)
% ### Your code comes here! ###
% Hint: Enter the observation that Earthquake = true
evidence = cell(1,N);
evidence{A} = true;
[engine, ll] = enter_evidence(engine, evidence);

m = marginal_nodes(engine, E);
disp(sprintf('p(E|A) = %g', m.T(true)));

% Fifth, compute P(E|B,A)
% ### Your code comes here! ###
% Hint: Enter the observation that Burglary = true and Alarm = true
evidence = cell(1,N);
evidence{B} = true;
evidence{A} = true;
[engine, ll] = enter_evidence(engine, evidence);

m = marginal_nodes(engine, E);
disp(sprintf('p(E|B,A) = %g', m.T(true)));

```

The answers by toolbox:

$$\begin{aligned}
p(A = t) &= 0.109 \\
p(B = t|A = t) &= 0.917431 \\
p(B = t|E = t, A = t) &= 0.1 \\
p(E = t|A = t) &= 0.0917431 \\
p(E = t|B = t, A = t) &= 0.01
\end{aligned}$$

- h. Second, we consider a more complex case. Assume that $P(B = t) = 0.1$, $P(E = t) = 0.01$ and $P(A | B, E)$ is described by the following CPT.

B, E	$P(A = f B, E), P(A = t B, E)$	
f, f	0.99	0.01
t, f	0.05	0.95
f, t	0.01	0.99
t, t	0.001	0.999

Compute $P(A = t), P(B = t|A = t), P(B = t|E = t, A = t), P(E = t|A = t), P(E = t|B = t, A = t)$ using the toolbox.

(Answer) The CPDs for this problem:

```

bnet.CPD{B} = tabular_CPD(bnet, B, [0.9 0.1]);
bnet.CPD{E} = tabular_CPD(bnet, E, [0.99 0.01]);
bnet.CPD{A} = tabular_CPD(bnet, A, [0.99 0.05 0.01 0.001 0.01 0.95 0.99 0.999]);

```

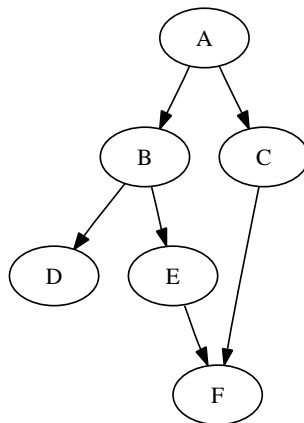
The answers by toolbox:

$$\begin{aligned}
p(A = t) &= 0.112869 \\
p(B = t|A = t) &= 0.842118 \\
p(B = t|E = t, A = t) &= 0.100817 \\
p(E = t|A = t) &= 0.087792 \\
p(E = t|B = t, A = t) &= 0.0105104
\end{aligned}$$

(Note that $P(B = t|A = t) > P(B = t|E = t, A = t)$ here.)

Problem 3: Bayes Nets, Computational Complexity

Use the following Bayes belief net over categorical variables when answering the parts of this problem:



- a. Take advantage of the independence assumptions in the Bayes belief net in order to factor and simplify the following expression for $P(B = b)$:

$$P(B = b) = \sum_{a \in A} \sum_{c \in C} \sum_{d \in D} \sum_{e \in E} \sum_{f \in F} P(a, b, c, d, e, f)$$

(Answer)

$$\begin{aligned} P(B = b) &= \sum_{a \in A} \sum_{c \in C} \sum_{d \in D} \sum_{e \in E} \sum_{f \in F} P(a, b, c, d, e, f) \\ &= \sum_{a \in A} \sum_{c \in C} \sum_{d \in D} \sum_{e \in E} \sum_{f \in F} P(a)P(b|a)P(c|a)P(d|b)P(e|b)P(f|c, e) \\ &= \sum_{a \in A} P(a)P(b|a) \sum_{c \in C} P(c|a) \sum_{d \in D} P(d|b) \sum_{e \in E} P(e|b) \sum_{f \in F} P(f|c, e) \\ &= \sum_{a \in A} P(a)P(b|a) \sum_{c \in C} P(c|a) \sum_{d \in D} P(d|b) \sum_{e \in E} P(e|b) \sum_{f \in F} P(f|c, e) \end{aligned}$$

Note that the sum

$$\sum_{c \in C} P(c|a) \sum_{d \in D} P(d|b) \sum_{e \in E} P(e|b) \sum_{f \in F} P(f|c, e) = 1 \quad (1)$$

so that we can write

$$P(B = b) = \sum_{a \in A} P(a)P(b|a) \quad (2)$$

- b. For this part assume that you have a computer that can perform multiplication and addition operations in one time step with memory operations taking no time. Assume also that the number of categories for each variable (A , B , C , D , E , and F) is k .

We now want to consider the computational complexity “big oh” expression for the order of the original *unsimplified* expression for $P(B = b)$ and your *simplified* expression for $P(B = b)$ in part (a), in terms of the number of addition and multiplication operations required to compute the answer.

Note that the “big oh” notation (so called because it uses the symbol O) describes the limiting behavior of a function for very small or very large arguments, usually in terms of simpler functions. Find out more information about this notation from

http://en.wikipedia.org/wiki/Big_O_notation or DHS book’s appendix.

For example, in the original unfactored formula for $P(B = b)$, there are $k^5 - 1$ additions, $O(k^5)$, while there are no multiplies, $O(1)$. Also, the total amount of space required to store the entire joint distribution, $P(A, B, C, D, E, F)$, naively is $k^6 - 1$, $O(k^6)$.

Now, using the same computer and category number assumptions, give the “big oh” expression for the order of your *simplified* expression for $P(B = b)$, again, in terms of the number of addition and multiplication operations required to compute the answer. Give the numbers of additions and multiplications that are necessary. Also, give the amount of space required to store the distributions.

(Answer)

The total number of additions, are $k - 1$, $O(k)$, and the total number of multiplications is also $k - 1$, $O(k)$. The amount of space required to store the distributions is $k - 1$ for $P(A)$ and $k * (k - 1)$ for $P(B|A)$, or $(k + 1)(k - 1)$, $O(k^2)$.

- c. Explain the complexity order relationship between the unsimplified expression of $P(B = b)$ and the simplified expression of $P(B = b)$. Why are they the same or different?

(Answer)

The unsimplified expression shows that the time complexity (number of additions and multiplications) is significantly higher than the time complexity of the simplified expression especially when the number of categories, k , becomes large. There is a memory complexity tradeoff in addition to the time complexity tradeoff, which results from factoring the complete joint probability table. Both of these complexity tradeoffs are a direct result of the strong independence assumptions inherent in the Bayes Belief Net.

- d. Now, in addition to the above Bayes belief net, you are told that there are only two categories for each variable, 0 (false) and 1 (true). You are also given these specific values for the conditional relationships:

$$\begin{aligned}
 P(A = 1) &= 1/2 \\
 P(B = 1|A) &= \begin{cases} 1/2 & \text{if } A=1 \\ 3/4 & \text{if } A=0 \end{cases} \\
 P(C = 1|A) &= \begin{cases} 1/4 & \text{if } A=1 \\ 1/2 & \text{if } A=0 \end{cases} \\
 P(D = 1|B) &= \begin{cases} 2/5 & \text{if } B=1 \\ 1/4 & \text{if } B=0 \end{cases} \\
 P(E = 1|B) &= \begin{cases} 2/3 & \text{if } B=1 \\ 4/5 & \text{if } B=0 \end{cases} \\
 P(F = 1|C, E) &= \begin{cases} 1/5 & \text{if } C=1 \text{ and } E=1 \\ 2/5 & \text{if } C=1 \text{ and } E=0 \\ 2/5 & \text{if } C=0 \text{ and } E=1 \\ 1/5 & \text{if } C=0 \text{ and } E=0 \end{cases}
 \end{aligned}$$

Using these specific values for the conditional distributions, calculate a numerical value for $P(B = 1)$ by hand.

(Answer)

$$P(B = 1) = \sum_{a \in A} P(a)P(B = 1|a) \tag{3}$$

$$= P(A = 1)P(B = 1|A = 1) + P(A = 0)P(B = 1|A = 0) \tag{4}$$

$$= \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{3}{4}\right) \tag{5}$$

$$= \frac{5}{8} \tag{6}$$

$$= 0.625 \tag{7}$$

- e. Using the same specific values for the conditional distributions in part (e), you are told that $F = 1$. Calculate $P(B = 1|F = 1)$ by hand.

(Answer)

For notational convenience, for a random variable X , we notate $X = 1$ by just X , and $X = 0$ by \bar{X} .

We will use the approach used in DHS 2.11 that avoids finding the value of the denominator in the Bayes relationship because it is more efficient in this binary state problem to calculate the numerator twice for both possible values. We can write the conditional probabilities for $P(B|F)$ and $P(\bar{B}|F)$ in terms of an extra constant α in each case:

$$P(B|F) = \frac{P(B, F)}{P(F)} \quad (8)$$

$$= \alpha P(B, F) \quad (9)$$

$$P(\bar{B}|F) = \frac{P(\bar{B}, F)}{P(F)} \quad (10)$$

$$= \alpha P(\bar{B}, F) \quad (11)$$

We can then use the relationship

$$P(B|F) + P(\bar{B}|F) = 1 \quad (12)$$

$$\alpha P(B, F) + \alpha P(\bar{B}, F) = 1 \quad (13)$$

$$\alpha [P(B, F) + P(\bar{B}, F)] = 1 \quad (14)$$

$$\alpha = \frac{1}{P(B, F) + P(\bar{B}, F)} \quad (15)$$

To solve for alpha and then for $P(B|F)$, we first find $P(B, F)$:

$$P(B, F) = \sum_{a \in A} P(a)P(B|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|B) \left[\sum_{e \in E} P(e|B)P(F|c, e) \right] \right] \right] \quad (16)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|B) [P(E|B)P(F|c, E) + P(\bar{E}|B)P(F|c, \bar{E})] \right] \right] \quad (17)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|B) \left[\frac{2}{3}P(F|c, E) + \frac{1}{3}P(F|c, \bar{E}) \right] \right] \right] \quad (18)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[\sum_{c \in C} P(c|a) \left[\frac{2}{3}P(F|c, E) + \frac{1}{3}P(F|c, \bar{E}) \right] \right] \quad (19)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[\begin{array}{c} P(C|a) \left[\frac{2}{3}P(F|C, E) + \frac{1}{3}P(F|C, \bar{E}) \right] \\ + \\ P(\bar{C}|a) \left[\frac{2}{3}P(F|\bar{C}, E) + \frac{1}{3}P(F|\bar{C}, \bar{E}) \right] \end{array} \right] \quad (20)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[\begin{array}{c} P(C|a) \left[\frac{2}{3} \frac{1}{5} + \frac{1}{3} \frac{2}{5} \right] \\ + \\ P(\bar{C}|a) \left[\frac{2}{3} \frac{2}{5} + \frac{1}{3} \frac{1}{5} \right] \end{array} \right] \quad (21)$$

$$= \sum_{a \in A} P(a)P(B|a) \left[P(C|a) \frac{4}{15} + P(\bar{C}|a) \frac{5}{15} \right] \quad (22)$$

$$= P(A)P(B|A) \left[P(C|A) \frac{4}{15} + P(\bar{C}|A) \frac{5}{15} \right] + P(\bar{A})P(B|\bar{A}) \left[P(C|\bar{A}) \frac{4}{15} + P(\bar{C}|\bar{A}) \frac{5}{15} \right] \quad (23)$$

$$= \frac{11}{22} \left[\frac{1}{4} \frac{4}{15} + \frac{3}{4} \frac{5}{15} \right] + \frac{13}{24} \left[\frac{1}{2} \frac{4}{15} + \frac{1}{2} \frac{5}{15} \right] \quad (24)$$

$$= \frac{23}{120} \approx 0.1917 \quad (25)$$

Next, we find $P(\bar{B}|F)$:

$$P(\bar{B}, F) = \sum_{a \in A} P(a)P(\bar{B}|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|\bar{B}) \left[\sum_{e \in E} P(e|\bar{B})P(F|c, e) \right] \right] \right] \quad (26)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|\bar{B}) [P(E|\bar{B})P(F|c, E) + P(\bar{E}|\bar{B})P(F|c, \bar{E})] \right] \right] \quad (27)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[\sum_{c \in C} P(c|a) \left[\sum_{d \in D} P(d|\bar{B}) \left[\frac{4}{5}P(F|c, E) + \frac{1}{5}P(F|c, \bar{E}) \right] \right] \right] \quad (28)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[\sum_{c \in C} P(c|a) \left[\frac{4}{5}P(F|c, E) + \frac{1}{5}P(F|c, \bar{E}) \right] \right] \quad (29)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[\begin{array}{c} P(C|a) \left[\frac{4}{5}P(F|C, E) + \frac{1}{5}P(F|C, \bar{E}) \right] \\ + \\ P(\bar{C}|a) \left[\frac{4}{5}P(F|\bar{C}, E) + \frac{1}{5}P(F|\bar{C}, \bar{E}) \right] \end{array} \right] \quad (30)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[\begin{array}{c} P(C|a) \left[\frac{4}{5} \frac{1}{5} + \frac{1}{5} \frac{2}{5} \right] \\ + \\ P(\bar{C}|a) \left[\frac{4}{5} \frac{2}{5} + \frac{1}{5} \frac{1}{5} \right] \end{array} \right] \quad (31)$$

$$= \sum_{a \in A} P(a)P(\bar{B}|a) \left[P(C|a) \frac{6}{25} + P(\bar{C}|a) \frac{9}{25} \right] \quad (32)$$

$$= P(A)P(\bar{B}|A) \left[P(C|A) \frac{6}{25} + P(\bar{C}|A) \frac{9}{25} \right] + P(\bar{A})P(\bar{B}|\bar{A}) \left[P(C|\bar{A}) \frac{6}{25} + P(\bar{C}|\bar{A}) \frac{9}{25} \right] \quad (33)$$

$$= \frac{11}{22} \left[\frac{1}{4} \frac{6}{25} + \frac{3}{4} \frac{9}{25} \right] + \frac{11}{24} \left[\frac{1}{2} \frac{6}{25} + \frac{1}{2} \frac{9}{25} \right] \quad (34)$$

$$= \frac{3}{25} \approx 0.12 \quad (35)$$

So, now we can use $P(B, F)$ and $P(\bar{B}, F)$ to find α :

$$\alpha = \frac{1}{P(B, F) + P(\bar{B}, F)} \quad (36)$$

$$\alpha \approx \frac{1}{0.1917 + 0.12} \quad (37)$$

$$\alpha \approx 3.2082 \quad (38)$$

And using α we can now solve for $P(B|F)$:

$$P(B|F) = \alpha P(B, F) \quad (39)$$

$$\approx (3.2082)(0.1917) \quad (40)$$

$$\approx 0.6150 \quad (41)$$

f. Compute $P(B = 1)$ and $P(B = 1|F = 1)$ by the Bayes Net Toolbox for Matlab.

(Answer)

The answers by toolbox: $p(B) = 0.625$, $p(B|F) = 0.614973$

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
%
% (for Problem 3(g))
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
N = 6;
dag = zeros(N,N);
A = 1; B = 2; C = 3; D = 4; E = 5; F = 6;
dag(A,B) = 1;
dag(A,C) = 1;
dag(B,D) = 1;
dag(B,E) = 1;
dag(E,F) = 1;
dag(C,F) = 1;

false= 1; true= 2;
ns = 2*ones(1,N); % binary nodes
names = {'A', 'B', 'C', 'D', 'E', 'F'};

% Set up the topology of the Bayes Net
bnet = mk_bnet(dag, ns, 'names', names, 'discrete', 1:N);

% Display the graph: download graph_draw from the link on the class webpage
graph_draw(bnet.dag, 'node_labels', names);

% Set up conditional probability tables for each node
bnet.CPD{A} = tabular_CPD(bnet, A, [1/2 1/2]);
bnet.CPD{B} = tabular_CPD(bnet, B, [1/4 1/2 3/4 1/2]);
bnet.CPD{C} = tabular_CPD(bnet, C, [1/2 3/4 1/2 1/4]);
bnet.CPD{D} = tabular_CPD(bnet, D, [3/4 3/5 1/4 2/5]);
bnet.CPD{E} = tabular_CPD(bnet, E, [1/5 1/3 4/5 2/3]);
bnet.CPD{F} = tabular_CPD(bnet, F, [4/5 3/5 3/5 4/5 1/5 2/5 2/5 1/5]);

% Choose an inference engine
engine = jtree_inf_engine(bnet);

% First, compute P(B=true)
evidence = cell(1,N);
[engine, ll] = enter_evidence(engine, evidence);

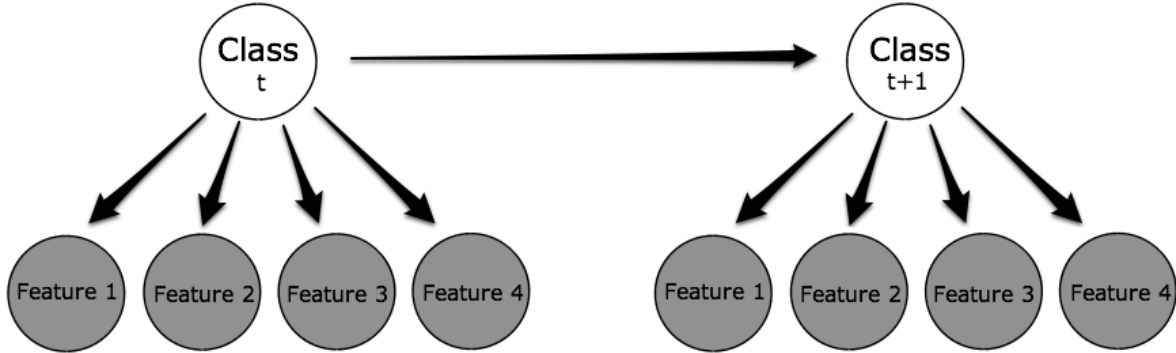
m = marginal_nodes(engine, B);
disp(sprintf('p(B) = %g', m.T(true)));

% Second, compute P(B=true|F=true)
evidence = cell(1,N);
evidence{F} = true;
[engine, ll] = enter_evidence(engine, evidence);

```

```
m = marginal_nodes(engine, B);
disp(sprintf('p(B|F) = %g', m.T(true)));
```

Problem 4: Dynamic Bayesian Networks



Now you are accustomed to modeling static Bayes nets. In this problem you play with a Dynamic Bayes Net (DBN) shown in the figure. This DBN has one discrete hidden (class) node and four discrete feature (observed) nodes per time slice. Again we use the Bayes Net Toolbox for Matlab. Please refer to the following tutorial page:

http://www.cs.ubc.ca/~murphyk/Bayes/usage_dbn.html

To help you solve this problem, we provide you with our Matlab example codes for the sample DBN (with one binary class node and two binary feature nodes) that we used in the lecture. Please download two Matlab files (“**generate_samples_dbn.m**”, “**learning_dbn.m**”, “**inference_dbn.m**” from the course website and run the codes to see how they works. Also, modify these codes to solve this problem. This will be easy, right?

The problem is decomposed into three phases. First, you model the DBN shown in the figure, and generate a toy dataset (1000 samples) from your modeled DBN (*Data Generation Phase*). Second, you divide the dataset into two parts (first 75% for training data and remaining 25% for testing data), and learn another new DBN based on the “training data” (*Learning (or Training) Phase*). Third, you test this learned DBN with the “testing data” (*Inference (or Testing) Phase*).

Although in this problem you deal with a toy dataset you generate in the data generation phase, you can use a similar approach (learning and inference) to train and test DBNs for target datasets in your research field. For your practical (research) purpose, the data generation phase in this problem could be replaced with your own research experiments that gather your target datasets.

- a. Define the DBN shown in the figure: one discrete hidden (class) node and four discrete feature (observed) nodes per time slice. Assume that the hidden class node and all feature nodes are binary (1 or 2). Then, generate 1000 samples (of two time slices) from the modeled DBN. Use the following probabilities for modeling the DBN, and modify “**generate_samples_dbn.m**”. (The given sample code models a DBN with one class node and two discrete feature nodes, so you have to change the code.) Write the printed CPT values. Also, explain the structures of the two variables in the code, a DBN sample instance “ev” and the corresponding evidence “cases{i}”.

C = Class node in time slice t ,

C' = Class node in time slice $t + 1$,

$F1$ = Feature 1 node, $F2$ = Feature 2 node,

$F3$ = Feature 3 node, $F4$ = Feature 4 node

Prior probabilities: $P(C = 1) = 0.5, P(C = 2) = 0.5$

Conditional probabilities: (*Warning: all the features take 1 or 2, not 0 or 1. This is also the convention for binary nodes in the toolbox.*)

$P(F1 = 1|C = 1) = 0.8, P(F1 = 1|C = 2) = 0.3$

$P(F2 = 1|C = 1) = 0.5, P(F2 = 1|C = 2) = 0.7$

$P(F3 = 1|C = 1) = 0.2, P(F3 = 1|C = 2) = 0.8$

$P(F4 = 1|C = 1) = 0.8, P(F4 = 1|C = 2) = 0.1$

State transition probabilities:

$P(C' = 1|C = 1) = 0.7, P(C' = 1|C = 2) = 0.2$

(Answer)

```
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
```

```
%
```

```
% Generate the samples from a specified DBN
```

```
%
```

```
% 9/23/2008: Adapted from Kevin Murphy's sample script
```

```
% by Hyungil Ahn (hiahn@media.mit.edu)
```

```
clear all
```

```
%%%
```

```
%% Do not change this seed part in this Data Generation phase
```

```
%% (With this, all the students can generate the same data samples)
```

```
seed = 0
```

```
rand('state', seed);
```

```
randn('state', seed);
```

```
%%%
```

```
N = 5; % the total number of nodes in each time slice
```

```
% Make into a Dynamic Bayes Net
```

```
% using the syntax from
```

```
% http://www.cs.ubc.ca/~murphyk/Bayes/usage\_dbn.html
```

```
%%
```

```
%intra topology is the same as the static case,
```

```
%the only inter connection is between the C and C' nodes
```

```
intra = zeros(N);
```

```
intra(1,2) = 1;
```

```
intra(1,3) = 1;
```

```
intra(1,4) = 1;
```

```
intra(1,5) = 1;
```

```
inter = zeros(N);
```

```
inter(1,1) = 1; % inter connection between the node 1's in time slice t and t+1
```

```

node_sizes = [2 2 2 2 2]; % the number of possible values that each node can take

onodes = 2:5; % observed nodes
dnodes = 1:5; % all the nodes per time slice

eclass1 = 1:5; % all the nodes per time slice
eclass2 = [6 2:5]; % here the node 6 refers the node 1 in time slice t+1
eclass = [eclass1 eclass2];

% make DBN for generating samples
% the generating DBN's name is dynBnet
dynBnet = mk_dbn(intra, inter, node_sizes, 'discrete', dnodes, ...
'eclass1', eclass1, 'eclass2', eclass2, ...
'observed', onodes);
for e=1:max(eclass)
    dynBnet.CPD{e} = tabular_CPD(dynBnet, e);
end

dynBnet.CPD{1} = tabular_CPD(dynBnet, 1, [0.5 0.5 ]); % prior P(C)
dynBnet.CPD{2} = tabular_CPD(dynBnet, 2, [0.8 0.3 0.2 0.7]); % conditional prob P(F1|C)
dynBnet.CPD{3} = tabular_CPD(dynBnet, 3, [0.5 0.7 0.5 0.3]); % conditional prob P(F2|C)
dynBnet.CPD{4} = tabular_CPD(dynBnet, 4, [0.2 0.8 0.8 0.2]); % conditional prob P(F3|C)
dynBnet.CPD{5} = tabular_CPD(dynBnet, 5, [0.8 0.1 0.2 0.9]); % conditional prob P(F4|C)
dynBnet.CPD{6} = tabular_CPD(dynBnet, 6, [0.7 0.2 0.3 0.8]); % state transition P(C'|C)

% print the CPT set up.
CPT = cell(1,N);
for i=1:N+1
    s=struct(dynBnet.CPD{i});
    CPT{i}=s.CPT;
end

for i=1:N+1
    sprintf('generating CPT for Node %d:', i)
    disp(cpt(CPT{i}))
end

%% since our nodes are discrete we can use the
%% dbn version of the jtree inference engine
engine = smoother_engine(jtree_2TBN_inf_engine(dynBnet));

E = length(engine);
ss = length(dynBnet.intra); % number of nodes per time slice
onodes = dynBnet.observed;

%%% Do not change this part (1000 samples of two time slices)
T = 2;
ncases = 1000;

```

```

%%
cases = cell(1, ncases);
cases_label = cell(1, ncases); % not used for learning

for i=1:ncases
    ev = sample_dbn(dynBnet, 'length', T);

    % used for building the confusion matrix in testing later, not used in learning
    cases_label{i} = ev(1,:);

    cases{i} = cell(ss,T);
    cases{i}(onodes,:) = ev(onodes, :);
end

% check the structure of ev, cases{i}

```

First, CPTs are as follows.

generating CPT for Node 1:

```

1 : 0.5000
2 : 0.5000

```

generating CPT for Node 2:

```

1 : 0.8000 0.2000
2 : 0.3000 0.7000

```

generating CPT for Node 3:

```

1 : 0.5000 0.5000
2 : 0.7000 0.3000

```

generating CPT for Node 4:

```

1 : 0.2000 0.8000
2 : 0.8000 0.2000

```

generating CPT for Node 5:

```

1 : 0.8000 0.2000
2 : 0.1000 0.9000

```

generating CPT for Node 6:

```

1 : 0.7000 0.3000
2 : 0.2000 0.8000

```

Second, for instance, the structures of the last (1000th) DBN sample instance “ev” and the corresponding evidence “cases{1000}” are as follows.

```

ev = [1][1]
      [1][1]
      [1][1]
      [2][1]
      [1][1]

```

```
cases{1000} = [ ] [ ]
              [1][1]
              [1][1]
              [2][1]
              [1][1]
```

A DBN sample evidence “ev” is a 5x2 cell object. The first and second columns of this object is for time slice t and time slice $t + 1$, respectively. Each column has the values of nodes $C, F1, F2, F3, F4$ in the sequence.

The variable “cases” have 1000 elements each of which is an evidence (5x2 cell) for learning. Here we showed the case corresponding to the last (1000th) evidence. Note that cases don’t have any values for the hidden class node (C and C'). In other words, the two cells in the first row of each case are empty ([] []). (They should be vacant. Also, this is the reason why we often have the swapped class IDs in learning.)

- b. Learn the parameters of a new DBN based on the training data (first 750 samples) generated in the previous phase. Modify “**learning_dbn.m**”. (Note that the given sample code uses all the generated samples (1000 samples) for learning the parameters, so you have to change the code.) Run many times varying the seed value of random number generation, and try to maximize the log likelihood in the EM steps. Write your seed value, the printed EM steps (with log likelihood values) and the obtained parameters (printed CPT values). Compare this estimated CPT values with the original CPT values. Also, explain the relationship among the number of iterations, the log likelihood and the accuracy of the obtained values.

```
%%%%%%%%%%
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
%
% Learn a new DBN based on the training data
%
% 9/23/2008: Adapted from Kevin Murphy’s sample script
%           by Hyungil Ahn (hiahn@media.mit.edu)

%%% You are allowed to change the seed number below.
%%% Please run several times varying the seed number to obtain the
%%% maximum log likelihood in the EM steps.
seed = 10
rand('state', seed);
randn('state', seed);
%%%

% make DBN for learning
% the learning DBN’s name is dynBnet2 (note that we should learn this without using
% any CPT information given to make dynBnet (the generating DBN))
dynBnet2 = mk_dbn(intra, inter, node_sizes, 'discrete', dnodes, ...
'eclass1', eclass1, 'eclass2', eclass2, ...
'observed', onodes);
for e=1:max(eclass)
    dynBnet2.CPD{e} = tabular_CPD(dynBnet2, e);
```

```

end
%

%% since our nodes are discrete we can use the
%% dbn version of the jtree inference engine
engine2 = smoother_engine(jtree_2TBN_inf_engine(dynBnet2));

% use the first 75% samples for training DBN
ntrain = ncases*0.75;

[dynBnet2, LL, engine2] = learn_params_dbn_em(engine2, cases(1:ntrain), 'max_iter', 20);

CPT = cell(1,N);
for i=1:N+1
    s=struct(dynBnet2.CPD{i}); % violate object privacy
    CPT{i}=s.CPT;
end

for i=1:N+1
    sprintf('learned CPT for Node %d:', i)
    disp(cpt(CPT{i}))
end

seed = 10

EM iteration 1, loglik = -6080.4932
EM iteration 2, loglik = -4105.3119
EM iteration 3, loglik = -4036.0039
EM iteration 4, loglik = -3939.1298
EM iteration 5, loglik = -3848.5071
EM iteration 6, loglik = -3808.4231
EM iteration 7, loglik = -3797.1704
EM iteration 8, loglik = -3793.5999

learned CPT for Node 1:
1 : 0.5033
2 : 0.4967

learned CPT for Node 2:
1 : 0.3023 0.6977
2 : 0.8116 0.1884

learned CPT for Node 3:
1 : 0.7029 0.2971
2 : 0.5099 0.4901

learned CPT for Node 4:

```

1 : 0.8132 0.1868

2 : 0.1992 0.8008

learned CPT for Node 5:

1 : 0.1203 0.8797

2 : 0.7598 0.2402

learned CPT for Node 6:

1 : 0.7751 0.2249

2 : 0.2484 0.7516

Here, in toolbox, class 1 and class 2 of the generating DBN (in (a)) was arbitrarily mapped to class 2 and class 1 of the learned DBN, respectively. So, the rows were swapped in Nodes 1, 2, 3, 4, 5. Also, for Node 6 (corresponding to $P(C'|C)$), both rows and columns were swapped (i.e., transposed). Thus, if the class IDs of the generating DBN had been equal to the class IDs of the learned DBN, the CPTs would be:

(matched)

learned CPT for Node 1:

1 : 0.4967

2 : 0.5033

learned CPT for Node 2:

1 : 0.8116 0.1884

2 : 0.3023 0.6977

learned CPT for Node 3:

1 : 0.5099 0.4901

2 : 0.7029 0.2971

learned CPT for Node 4:

1 : 0.1992 0.8008

2 : 0.8132 0.1868

learned CPT for Node 5:

1 : 0.7598 0.2402

2 : 0.1203 0.8797

learned CPT for Node 6:

1 : 0.7516 0.2484

2 : 0.2249 0.7751

The CPTs in the learned DBN is very similar to those in the original DBN. We see that as the number of iterations increase, the log likelihood and the accuracy of the obtained values also increase.

Additional notes:

When we use seed = 250,

EM iteration 1, loglik = -6129.3068

EM iteration 2, loglik = -4172.9729

EM iteration 3, loglik = -4122.3517

EM iteration 4, loglik = -4105.5025

EM iteration 5, loglik = -4090.8458

EM iteration 6, loglik = -4068.0639

EM iteration 7, loglik = -4029.6295
EM iteration 8, loglik = -3971.6066
EM iteration 9, loglik = -3900.5234
EM iteration 10, loglik = -3840.4228
EM iteration 11, loglik = -3809.4375
EM iteration 12, loglik = -3798.2680
EM iteration 13, loglik = -3794.4239
EM iteration 14, loglik = -3792.8028

learned CPT for Node 1:

1 : 0.5007
2 : 0.4993

learned CPT for Node 2:

1 : 0.8115 0.1885
2 : 0.2995 0.7005

learned CPT for Node 3:

1 : 0.5090 0.4910
2 : 0.7049 0.2951

learned CPT for Node 4:

1 : 0.2069 0.7931
2 : 0.8091 0.1909

learned CPT for Node 5:

1 : 0.7595 0.2405
2 : 0.1170 0.8830

learned CPT for Node 6:

1 : 0.7636 0.2364
2 : 0.2121 0.7879

- c. Now you use the portion of the data (remaining 250 samples) set aside for testing to test the obtained classifier in the previous phase. Modify “**inference_dbn.m**”. (Note that the given sample code uses all the generated samples (1000 samples) for testing the classifier, so you have to change the code.) Show the confusion matrix of the results using a DBN junction tree inference engine. What is the test accuracy?

For the learned DBN (using seed = 10), the confusion matrix (confM) is $\begin{bmatrix} 14 & 108 \\ 119 & 9 \end{bmatrix}$

Note that the columns in the above confusion matrix should be swapped if the class IDs of the generating DBN had been matched to the class IDs of the learned DBN. Thus, the confusion matrix is: $\begin{bmatrix} 108 & 14 \\ 9 & 119 \end{bmatrix}$

Classification accuracy is $(108+119)/250 = 0.908$ (90.8%)

Additional notes:

For the learned DBN (using seed = 250), the confusion matrix (confM) is $\begin{bmatrix} 108 & 14 \\ 9 & 119 \end{bmatrix}$

Classification accuracy is $(108+119)/250 = 0.908$ (90.8%)

```

% MIT MAS622j/1.126J Fall 2008, Problem Set 3
%
% Test the learned DBN for the testing data
%
% 9/23/2008: Adapted from Kevin Murphy's sample script
%           by Hyungil Ahn (hiahn@media.mit.edu)

% Confusion Matrix
% Here, the class node are binary, so the confM is 3x3 matrix

confM = zeros(2)

for i=ntrain+1:ncases % for remaining 25% samples

    evidence = cell(ss,T);
    evidence(onodes,:) = cases{i}(onodes,:);

    % engine2 is the learned DBN. give evidence
    engine2 = enter_evidence(engine2, evidence);

    % referring to 1st node (hidden class node) in 2nd time slice (t+1)
    m = marginal_nodes(engine2, 1, 2);

    % referring to the true class information of the hidden class node in
    % 2nd time slice
    truth = cases_label{i}{1,2};
    inferredClass = argmax(m.T);
    confM(truth,inferredClass) = confM(truth,inferredClass) + 1;

end
%
confM
bar3(confM)

```

- d. Using the same generated data, do the learning and testing phases again. But this time use the first 250 samples for testing and the last 750 samples for learning. Show your seed value, the estimated CPT values, the confusion matrix, and the accuracy.

The above codes should be changed as follows:

In learning,

```

ntrain = ncases*0.75;
[dynBnet2, LL, engine2] = learn_params_dbn_em(engine2, cases(ncases-ntrain+1:ncases), ...
'max_iter', 20); % for the last 75% samples

```

In inference,

```

confM = zeros(2)

for i=1:ncases-ntrain % for remaining first 25% samples

```

```

evidence = cell(ss,T);
evidence(onodes,:) = cases{i}(onodes,:);

engine2 = enter_evidence(engine2, evidence); % engine2 is the learned DBN. give evidence

% referring to 1st node (hidden class node) in 2nd time slice (t+1)
m = marginal_nodes(engine2, 1, 2);

% referring to the true class information of the hidden class node in 2nd time slice
truth = cases_label{i}{1,2};

inferredClass = argmax(m.T);
confM(truth,inferredClass) = confM(truth,inferredClass) + 1;

end

```

For the learned DBN (using seed = 10),

```

EM iteration 1, loglik = -6143.3171
EM iteration 2, loglik = -4107.7576
EM iteration 3, loglik = -4036.1195
EM iteration 4, loglik = -3929.9935
EM iteration 5, loglik = -3827.2723
EM iteration 6, loglik = -3784.2889
EM iteration 7, loglik = -3773.7251
EM iteration 8, loglik = -3770.7456

```

learned CPT for Node 1:

```

1 : 0.4650
2 : 0.5350

```

learned CPT for Node 2:

```

1 : 0.2760 0.7240
2 : 0.8108 0.1892

```

learned CPT for Node 3:

```

1 : 0.7311 0.2689
2 : 0.5068 0.4932

```

learned CPT for Node 4:

```

1 : 0.8037 0.1963
2 : 0.1932 0.8068

```

learned CPT for Node 5:

```

1 : 0.1112 0.8888
2 : 0.7540 0.2460

```

learned CPT for Node 6:

```

1 : 0.7825 0.2175
2 : 0.2925 0.7075

```

The confusion matrix (confM) is $\begin{bmatrix} 16 & 105 \\ 112 & 17 \end{bmatrix}$

Classification accuracy is $(105+112)/250 = 0.8680$ (86.8%)

Additional notes:

For the learned DBN (using seed = 250),

EM iteration 1, loglik = -6140.5740
EM iteration 2, loglik = -4171.5210
EM iteration 3, loglik = -4116.7681
EM iteration 4, loglik = -4094.0495
EM iteration 5, loglik = -4067.0760
EM iteration 6, loglik = -4021.4153
EM iteration 7, loglik = -3951.9729
EM iteration 8, loglik = -3871.1412
EM iteration 9, loglik = -3808.8016
EM iteration 10, loglik = -3780.8328
EM iteration 11, loglik = -3772.5740
EM iteration 12, loglik = -3770.3022

learned CPT for Node 1:

1 : 0.5377
2 : 0.4623

learned CPT for Node 2:

1 : 0.8103 0.1897
2 : 0.2748 0.7252

learned CPT for Node 3:

1 : 0.5049 0.4951
2 : 0.7338 0.2662

learned CPT for Node 4:

1 : 0.2005 0.7995
2 : 0.7982 0.2018

learned CPT for Node 5:

1 : 0.7566 0.2434
2 : 0.1064 0.8936

learned CPT for Node 6:

1 : 0.7108 0.2892
2 : 0.2121 0.7879

The confusion matrix (confM) is

$$\begin{bmatrix} 105 & 16 \\ 17 & 112 \end{bmatrix}$$

Classification accuracy is $(105+112)/250 = 0.8680$ (86.8%)

Prediction is very difficult, especially about the future. - Niels Bohr