

Probabilistic models, Spring 2013
Exercise session 4: Solutions

Note: The source code snippets which were used to solve these exercises should run both in Octave and MATLAB.

Note2: In the source code $*$ is a matrix multiplication (that is, matrix-by-matrix, matrix-by-vector or vector-by-matrix) if the multiplicands are matrices or vectors, while $.*$ is an elementwise multiplication. Especially in assignment 17–18 matrix multiplication is used heavily.

14. a) For each of 16 cases (x_1, x_2, x_3, x_4) we get the result by first computing

$$P(Y | x_1, x_2, x_3, x_4) \propto P(Y)P(x_1 | Y)P(x_2 | Y)P(x_3 | Y)P(x_4 | Y)$$

and then normalizing the three probabilities to sum up to 1.

Resulting 16 distributions:

X_1	X_2	X_3	X_4	$Y = 1$	$Y = 2$	$Y = 3$
0	0	0	0	0.7678	0.1991	0.0332
0	0	0	1	0.9897	0.0071	0.0032
0	0	1	0	0.1513	0.8235	0.0252
0	0	1	1	0.8594	0.1300	0.0106
0	1	0	0	0.0443	0.9299	0.0258
0	1	0	1	0.6148	0.3586	0.0266
0	1	1	0	0.0023	0.9927	0.0051
0	1	1	1	0.0745	0.9130	0.0124
1	0	0	0	0.6168	0.0457	0.3376
1	0	0	1	0.9591	0.0020	0.0389
1	0	1	0	0.2143	0.3333	0.4524
1	0	1	1	0.8336	0.0360	0.1304
1	1	0	0	0.0695	0.4170	0.5135
1	1	0	1	0.5834	0.0972	0.3193
1	1	1	0	0.0064	0.8103	0.1833
1	1	1	1	0.1513	0.5294	0.3193

- b) For $Y = y$, the condition distribution is

$$P(X_1, X_2, X_3, X_4 | y) = P(X_1 | y)P(X_2 | y)P(X_3 | y)P(X_4 | y).$$

Resulting 3 distributions:

X_1	X_2	X_3	X_4	$Y = 1$	$Y = 2$	$Y = 3$
0	0	0	0	0.0324	0.0168	0.0084
0	0	0	1	0.2916	0.0042	0.0056
0	0	1	0	0.0036	0.0392	0.0036
0	0	1	1	0.0324	0.0098	0.0024
0	1	0	0	0.0036	0.1512	0.0126
0	1	0	1	0.0324	0.0378	0.0084
0	1	1	0	0.0004	0.3528	0.0054
0	1	1	1	0.0036	0.0882	0.0036
1	0	0	0	0.0486	0.0072	0.1596
1	0	0	1	0.4374	0.0018	0.1064
1	0	1	0	0.0054	0.0168	0.0684
1	0	1	1	0.0486	0.0042	0.0456
1	1	0	0	0.0054	0.0648	0.2394
1	1	0	1	0.0486	0.0162	0.1596
1	1	1	0	0.0006	0.1512	0.1026
1	1	1	1	0.0054	0.0378	0.0684

c) The joint distribution for X is given by marginalizing over Y :

$$P(X_1, X_2, X_3, X_4) = \sum_y P(Y = y)P(X_1, X_2, X_3, X_4 | Y = y)$$

Resulting distribution:

X_1	X_2	X_3	X_4	$P(X_1, X_2, X_3, X_4)$
0	0	0	0	0.0253
0	0	0	1	0.1768
0	0	1	0	0.0143
0	0	1	1	0.0226
0	1	0	0	0.0488
0	1	0	1	0.0316
0	1	1	0	0.1066
0	1	1	1	0.0290
1	0	0	0	0.0473
1	0	0	1	0.2736
1	0	1	0	0.0151
1	0	1	1	0.0350
1	1	0	0	0.0466
1	1	0	1	0.0500
1	1	1	0	0.0560
1	1	1	1	0.0214

d) We can use the method given in lecture slides, that is, compute

$$P(X_1 | X_2 = 0, X_3 = 0) \propto \sum_y P(Y = y)P(X_1 | Y = y)P(X_2 | Y = y)P(X_3 | Y = y)$$

and then normalize the results. This way we can avoid dealing with X_4 altogether.

Alternatively we can just sum over the results in c) using the definition of conditional probability:

$$P(X_1 | X_2 = 0, X_3 = 0) = \frac{P(X_1, X_2 = 0, X_3 = 0)}{P(X_2 = 0, X_3 = 0)} = \frac{\sum_{x_4} P(X_1, X_2 = 0, X_3 = 0, X_4 = x_4)}{\sum_{x_1, x_4} P(X_1 = x_1, X_2 = 0, X_3 = 0, X_4 = x_4)}$$

Either way, we get:

$$P(X_1 = 0 | X_2 = 0, X_3 = 0) = 0.3864$$

$$P(X_1 = 1 | X_2 = 0, X_3 = 0) = 0.6136$$

Code that computes all the above probabilities:

```
#!/usr/bin/octave -q

% conditional probabilities
PY = [.6, .3, .1]; % P(Y)
PX1gY = [.4, .6; .7, .3; .05, .95]; % P(X_1 | Y)
PX2gY = [.9, .1; .1, .9; .4, .6]; % P(X_2 | Y)
PX3gY = [.9, .1; .3, .7; .7, .3]; % P(X_3 | Y)
PX4gY = [.1, .9; .8, .2; .6, .4]; % P(X_4 | Y)

fprintf('\na) classification distribution\n\n');
% for each X
for x1 = 0:1, for x2 = 0:1, for x3 = 0:1, for x4 = 0:1
    PYgx = PY .* PX1gY(:, x1+1)' .* PX2gY(:, x2+1)' .* PX3gY(:, x3+1)' .* PX4gY(:, x4+1)';
    PYgx = PYgx / sum(PYgx);
    fprintf('P(Y | X=[%d, %d, %d, %d]) = [%.4f, %.4f, %.4f]\n', x1, x2, x3, x4, PYgx);
end, end, end, end

fprintf('\nb) conditional distributions\n');
% for each Y
for y = 0:2
```

```

fprintf('\nY=%d:\n', y);
% for each X
for x1 = 0:1, for x2 = 0:1, for x3 = 0:1, for x4 = 0:1
    Pxgy = PX1gY(y+1,x1+1) .* PX2gY(y+1,x2+1) .* PX3gY(y+1,x3+1) .* PX4gY(y+1,x4+1);
    fprintf(' P(X=[%d, %d, %d, %d] | Y=%d) = %.4f\n', x1, x2, x3, x4, y, Pxgy);
end, end, end, end
end

fprintf('\nc) marginal distribution for X\n\n');
% for each X
for x1 = 0:1, for x2 = 0:1, for x3 = 0:1, for x4 = 0:1
    Px = sum(PY .* PX1gY(:,x1+1)' .* PX2gY(:,x2+1)' .* PX3gY(:,x3+1)' .* PX4gY(:,x4+1)');
    fprintf(' P(X=[%d, %d, %d, %d]) = %.4f\n', x1, x2, x3, x4, Px);
end, end, end, end

fprintf('\nd) \n\n');
PX1gx2x3 = PY .* PX2gY(:,0+1)' .* PX3gY(:,0+1)' * PX1gY(:,:);
PX1gx2x3 = PX1gx2x3 / sum(PX1gx2x3);
fprintf(' P(X1=0 | X2=0, X3=0) = %.4f\n', PX1gx2x3(1));
fprintf(' P(X1=1 | X2=0, X3=0) = %.4f\n', PX1gx2x3(2));

```

15.–16. We treat the observation pair (X, Y) as a single variable O . Since X and Y are independent given the location L , the conditional probabilities for an observation $o = (x, y)$ is given by:

$$P(O = (x, y) | L) = P(X = x, Y = y | L) = P(X = x | L)P(Y = y | L)$$

a) Smoothed distributions are computed using the forward–backward algorithm.

In the forward algorithm we have the following recursive definition:

$$P(L_{t+1} | x_{1:t+1}, y_{1:t+1}) \propto P(x_{t+1} | L_{t+1})P(y_{t+1} | L_{t+1}) \sum_{l_t} P(L_{t+1} | l_t)P(l_t | x_{1:t}, y_{1:t})$$

We get the following results for $P(L_t | x_{1:t}, y_{1:t})$:

t	$L_t = A$	$L_t = B$	$L_t = C$	$L_t = D$
1	0.2000	0.2250	0.3750	0.2000
2	0.1245	0.3891	0.3619	0.1245
3	0.0149	0.0979	0.1562	0.7311
4	0.0165	0.3109	0.2493	0.4234
5	0.3197	0.3568	0.2456	0.0779
6	0.1408	0.4607	0.2946	0.1039

And in the backward algorithm the following:

$$P(x_{t+1:6}, y_{t+1:6} | L_t) = \sum_{l_{t+1}} P(l_{t+1} | L_t)P(x_{t+1} | l_{t+1})P(y_{t+1} | l_{t+1})P(x_{t+2:6}, y_{t+2:6} | l_{t+1})$$

We get the following results for $P(x_{t+1:6}, y_{t+1:6} | L_t)$:

t	$L_t = A$	$L_t = B$	$L_t = C$	$L_t = D$
5	0.106667	0.080000	0.066667	0.106667
4	0.010178	0.008889	0.007689	0.005911
3	0.000654	0.000699	0.000574	0.000862
2	0.000052	0.000164	0.000172	0.000191
1	0.000016	0.000012	0.000010	0.000018

Given the forward and backward probabilities, the smoothed probabilities are given by:

$$P(L_t | x_{1:6}, y_{1:6}) \propto P(L_t | x_{1:t}, y_{1:t})P(x_{t+1:6}, y_{t+1:6} | L_t)$$

We get the following final results for $P(L_t | x_{1:6}, y_{1:6})$:

t	$L_t = A$	$L_t = B$	$L_t = C$	$L_t = D$
1	0.2438	0.2015	0.2832	0.2715
2	0.0414	0.4087	0.3980	0.1519
3	0.0122	0.0857	0.1124	0.7896
4	0.0229	0.3759	0.2607	0.3405
5	0.3905	0.3268	0.1875	0.0951
6	0.1408	0.4607	0.2946	0.1039

- b) We use Viterbi algorithm to compute the most probable sequence $\arg \max_{L_{1:6}} P(L_{1:6} | x_{1:6}, y_{1:6})$. By defining $p(L, t) = \max_{l_{1:t-1}} P(l_{1:t-1}, L | x_{1:t}, y_{1:t})$ we get the following recursive formula:

$$p(L, t) = P(x_t, y_t | L) \max_{l'} p(l', t-1) P(L | l')$$

Now $\max_l p(l, 6)$ gives us the probability of the most probable sequence.

To find out the actual sequence we use backtracking. While computing $p(L, t)$ for each L , we also save the previous state l' that maximizes the probability of the sequence given that the t th state is L . The most probable sequence can then be found by travelling backwards these states.

The most probably path given by the algorithm is $C \rightarrow C \rightarrow D \rightarrow B \rightarrow A \rightarrow B$. (Actually there are two different sequences with the highest probability, the other one is $C \rightarrow C \rightarrow D \rightarrow D \rightarrow B \rightarrow B$.)

The posterior probability of a path $L_{1:6}$ is

$$P(L_{1:6} | x_{1:6}, y_{1:6}) = \frac{P(L_{1:6}) P(x_{1:6}, y_{1:6} | L_{1:6})}{P(x_{1:6}, y_{1:6})},$$

where

$$\begin{aligned} P(L_{1:6}) &= \prod_{t=1}^6 P(L_t | L_{1:t-1}) \\ &= \prod_{t=2}^6 P(L_t | L_{t-1}) \cdot \sum_{l_0} P(L_1 | l_0) P(l_0) \end{aligned}$$

and

$$P(x_{1:6}, y_{1:6} | L_{1:6}) = \prod_t P(x_t | L_t) P(y_t | L_t).$$

The denominator is a bit more complicated, but it can be calculated as a part of the Viterbi recursion. We want to compute $P(o_{1:6}) = \sum_{l_6} P(o_{1:6}, l_6)$, where the term inside the sum can be defined recursively:

$$\begin{aligned} P(o_{1:t}, l_t) &= P(o_t | l_t) P(o_{1:t-1}, l_t) \\ &= P(o_t | l_t) \sum_{l_{t-1}} P(l_t, l_{t-1}) P(o_{1:t-1}, l_{t-1}). \end{aligned}$$

So similarly to $p(L, t)$ we define $q(L, t) = P(o_{1:t}, L_t = L)$ and get the recursive formula

$$q(L, t) = P(x_t, y_t | L) \sum_{l'} q(l', t-1) P(L | l').$$

The only difference to the Viterbi recursion is having sum instead of maximization.

Since the number of variables is small, in this case it is also possible to just sum over all paths:

$$P(x_{1:6}, y_{1:6}) = \sum_{l_{1:6}} P(x_{1:6}, y_{1:6} | l_{1:6}) P(l_{1:6}).$$

Either way, we get the posterior probability of the most probable path:

$$P(L_{1:6} = (C, C, D, B, A, B) | x_{1:6}, y_{1:6}) \approx 0.018.$$

Code:

```
#!/usr/bin/octave -q

% values are encoded as follows:
% L: A=1, B=2, C=3, D=4
% X,Y: low=1, medium=2, high=3

% transition probabilities:  $PLtr(i,j) = P(L_{t+1} = j \mid L_t = i)$ 
PLtr = [1 1 1 0; 1 1 0 1; 1 0 1 1; 0 1 1 1] / 3;
% emission probabilities:  $PXgL(i,x) = P(X = x \mid L = i)$ 
PXgL = [1 2 7; 3 4 3; 3 4 3; 7 2 1] / 10;
PYgL = [7 2 1; 3 4 3; 2 3 5; 1 2 7] / 10;

% prior probabilities
PL0 = [2 3 3 2] / 10;

% observations
x = [3 2 1 1 3 2];
y = [3 2 3 2 2 2];

fprintf('\na) smoothed distributions: \n');

% forward algorithm: compute  $PLf(t+1,i) = P(L_{t+1} = i \mid x_{1:t}, y_{1:t})$ 
fprintf('\nforward algorithm probabilities:\n');
PLf = zeros(7, 4);
PLf(1, :) = PL0; % prior probabilities
for t = 1:6
    propto = PXgL(:,x(t))' .* PYgL(:,y(t))' .* (PLtr' * PLf(t,:))';
    PLf(t+1, :) = propto / sum(propto);
    fprintf(' P(L_%d | x_{1:%d}, y_{1:%d}) = [%4f, %4f, %4f, %4f]\n', ...
        t, t, t, PLf(t+1,:));
end

% backward algorithm: compute  $PLb(t,i) = P(x_{t+1:6}, y_{t+1:6} \mid L_t = i)$ 
fprintf('\nbackward algorithm probabilities:\n');
PLb = zeros(6, 4);
PLb(6, :) = 1; % with this "trick" we don't need a special case handling for the first step
for t = 5:-1:1
    PLb(t, :) = (PLtr * (PXgL(:,x(t+1)) .* PYgL(:,y(t+1)) .* PLb(t+1,:)))';
    fprintf(' P(x_{%d:6}, y_{%d:6} | L_%d) = [%6f, %6f, %6f, %6f]\n', ...
        t+1, t+1, t, PLb(t,:));
end

% smoothed probabilities
fprintf('\nsmoothed probabilities:\n');
for t = 1:6
    PL = PLf(t+1,:) .* PLb(t,:);
    PL = PL / sum(PL);
    fprintf(' P(L_%d | x_{1:6}, y_{1:6}) = [%4f, %4f, %4f, %4f]\n', t, PL);
end

fprintf('\n\nb) the most probable path: \n\n');

% viterbi algorithm: compute  $p(i,t) = \max_{l_{1:t-1}} P(l_{1:t-1}, L_t=i \mid x_{1:t}, y_{1:t})$ 
p = zeros(4, 6); % the probability of the most probable path (up to a constant factor)
q = zeros(4, 6); % the probability of all paths
b = zeros(4, 6); % previous state of the most probable path
p(:,1) = PXgL(:,x(1)) .* PYgL(:,y(1)) .* (PL0 * PLtr)'; %  $p(i,1) = P(L_t=i \mid x_1, y_1)$ 
q(:,1) = PXgL(:,x(1)) .* PYgL(:,y(1)) .* (PL0 * PLtr)'; %  $q(i,1) = P(L_t=i \mid x_1, y_1)$ 
for t = 2:6
    [pp,pb] = max( repmat(p(:,t-1), 1, 4) .* PLtr, [], 1);
    p(:,t) = PXgL(:,x(t)) .* PYgL(:,y(t)) .* pp';
    b(:,t) = pb';
    q(:,t) = PXgL(:,x(t)) .* PYgL(:,y(t)) .* (q(:,t-1)' * PLtr)';
end

% backtracking
```

```

l = zeros(6, 1);
[maxp, l(6)] = max(p(:,6));
for t = 5:-1:1
    l(t) = b(l(t+1), t+1);
end

fprintf('The sequence l with highest probability: %d -> %d -> %d -> %d -> %d -> %d\n', l);

% compute P(l)
Pl = sum(PLtr(:, l(1)))' .* PL0;
for t = 2:6
    Pl = Pl * PLtr(l(t-1), l(t));
end

% compute P(x, y | l)
Pxygl = 1;
for t = 1:6
    Pxygl = Pxygl * PXgL(l(t), x(t)) * PYgL(l(t), y(t));
end

% compute P(x, y)
Pxy = sum(q(:, 6));

% compute P(l | x, y)
Plgxy = Pxygl * Pl / Pxy;

fprintf('\n\nThe probability of the sequence l: P(l | x, y) = %g\n', Plgxy);

```

17.–18. Using belief propagation as described in the slides we get the following marginal posterior probabilities for evidence $A = 0, H = 0$:

	$x = 0$	$x = 1$
$P(B = x A = 0, H = 0)$	0.6376	0.3624
$P(C = x A = 0, H = 0)$	0.5247	0.4753
$P(D = x A = 0, H = 0)$	0.8728	0.1272
$P(E = x A = 0, H = 0)$	0.6254	0.3746
$P(F = x A = 0, H = 0)$	0.8966	0.1034
$P(G = x A = 0, H = 0)$	0.4310	0.5690

First, the λ messages for all nodes are computed, then the π messages, and then the beliefs are calculated as their product.

Code:

```

#!/usr/bin/octave -q

% conditional probabilities
PA = [.3 .7];
PDgA = [.9 .1; .7 .3];
PEgD = [.6 .4; .8 .2];
PCgE = [.3 .7; .9 .1];
PBgC = [.4 .6; .9 .1];
PFgD = [.5 .5; .7 .3];
PGgF = [.4 .6; .7 .3];
PHgF = [.8 .2; .1 .9];

% notation:
% piY = \pi(Y), lambdaY = \lambda(Y)
% piXY = \pi_X(Y), lambdaXY = \lambda_X(Y)
% PXgah = P(X | a, h)

% initial pi and lambda
piA = [1 0]; % evidence
lambdaA = [1 0]'; % evidence

```

```

piH = [1 0];          % evidence
lambdaH = [1 0]';    % evidence
lambdaG = [1 1]';    % no children
lambdaB = [1 1]';    % no children

% belief propagation
lambdaBC = PBgC * lambdaB
lambdaC = lambdaBC

lambdaCE = PCgE * lambdaC
lambdaE = lambdaCE

lambdaGF = PGgF * lambdaG
lambdaHF = PHgF * lambdaH
lambdaF = lambdaGF .* lambdaHF

lambdaED = PEgD * lambdaE
lambdaFD = PFgD * lambdaF
lambdaD = lambdaED .* lambdaFD

piDA = piA
piD = piDA * PDgA

piED = lambdaFD' .* piD
piE = piED * PEgD

piCE = piE
piC = piCE * PCgE

piBC = piC
piB = piBC * PBgC

piFD = lambdaED' .* piD
piF = piFD * PFgD

piGF = lambdaHF' .* piF
piG = piGF * PGgF

% final beliefs
belB = piB .* lambdaB';
belC = piC .* lambdaC';
belD = piD .* lambdaD';
belE = piE .* lambdaE';
belF = piF .* lambdaF';
belG = piG .* lambdaG';

% probabilities by normalization
PBgah = belB / sum(belB);
PCgah = belC / sum(belC);
PDgah = belD / sum(belD);
PEgah = belE / sum(belE);
PFgah = belF / sum(belF);
PGgah = belG / sum(belG);

fprintf('P(B | a,h) = [%.4f, %.4f]\n', PBgah);
fprintf('P(C | a,h) = [%.4f, %.4f]\n', PCgah);
fprintf('P(D | a,h) = [%.4f, %.4f]\n', PDgah);
fprintf('P(E | a,h) = [%.4f, %.4f]\n', PEgah);
fprintf('P(F | a,h) = [%.4f, %.4f]\n', PFgah);
fprintf('P(G | a,h) = [%.4f, %.4f]\n', PGgah);

```