CPSC 540 Assignment 4 (due March 20)

Graphical Models and Paper Review

1 Markov Chains

1.1 Sampling, Inference, and Decoding

The function *example_markovChain.m* loads the initial state probabilities and transition probabilities for three Markov chain models on d binary variables,

$$p(x_1, x_2, \dots, x_d) = p(x_1) \prod_{j=2}^{d} p(x_j | x_{j-1}).$$

It then tries to find the optimal decoding (the most likely assignment to the variables $\{x_1, x_2, \dots, x_d\}$) in each of the three chains. In the demo, decoding is done by enumerating all possible assignments to the variables. This works for the first two chains as they only have 4 variables, but is too slow on the last chain because it has 31 variables. In this question you'll explore two ways to estimate the marginals in third Markov chain and two ways to estimate the most-probable sequence. You only need to report results on the long Markov chain for this question.

1. Write a function, *sampleAncestral.m*, that uses ancestral sampling to sample sequence x. Hand in this code and report all the univariate marginal probabilities using a Monte Carlo estimate based on 10000 samples.
2. Write a function, *marginalCK.m*, that uses the CK equations to compute the exact univariate marginals. Hand in this code and report all exact univariate marginals.
3. Write a function, *marginalDecode.m*, that returns the sequence of states x_j that maximize the marginal probability $p(x_j)$ (for each j). Hand in this code and report the sequence of most likely states.
4. Write a function, *viterbiDecode.m*, that implements the Viterbi decoding algorithm for Markov chains. Hand in this code and report the optimal decoding of the third Markov chain.

Hint: for parts 2-4, you can use a 2 by d matrix M to represent the dynamic programming table, and for part 4 you can use another matrix B containing the argmax values that lead to each entry in the table.

Answer:

1. The code could look roughly like this:

```
function [X] = sampleAncestral(p0,pT,nSamples)
nNodes = size(pT,3)+1;

for i = 1:nSamples
    if rand < p0(1)
        X(i,1) = 1;
    else
        X(i,1) = 2;
    end
    for j = 2:nNodes
       if rand < pT(X(i,j-1),1,j-1)
           X(i,j) = 1;
       else
           X(i,j) = 2;
       end
    end
end
```

On one run, the marginal estimates for state 1 were given by: 0.5966 0.4671 0.9031 0.9704 0.7827 0.2258 0.0533 0.3380 0.3118 0.5888 0.3628 0.7406 0.3649 0.1295 0.3758 0.4134 0.7226 0.7897 0.3095 0.6704 0.5833 0.2601 0.4011 0.2412 0.7348 0.6201 0.6062 0.7467 0.4704 0.4367 0.1774.

2. The code could look roughly like this:

```
function [V] = marginalCK(p0,pT)
nNodes = size(pT,3)+1;

V = zeros(2,nNodes);
% Fill Dynamic Programming table
V(:,1) = p0;
[~,yOpt(1)] = max(V(:,1));
for t = 2:nNodes
    V(:,t) = pT(:,:,t-1)'*V(:,t-1);
end
```

The exact marginals are given by 0.6000 0.4681 0.8998 0.9718 0.7779 0.2267 0.0542 0.3359 0.3204 0.5902 0.3601 0.7479 0.3726 0.1313 0.3744 0.4103 0.7240 0.7877 0.3147 0.6698 0.5811 0.2541 0.4037 0.2403 0.7378 0.6211 0.6060 0.7419 0.4644 0.4380 0.1782. These values agree with the Monte Carlo estimate up to the 2nd decimal place.

3. The code should look similar to the above, but where you compute the max of each row of V. The sequence of marginal most-likely states is: 1 2 1 1 1 2 2 2 2 1 2 1 2 2 2 2 1 1 2 1 1 2 2 2 1 1 1 1 2 2 2.

4. The code should look roughly like this:

```
function [yOpt] = viterbiDecode(p0,pT)
% Exact decoding of Markov binary chains

nNodes = size(pT,3)+1;

V = zeros(2,nNodes);
T = zeros(2,nNodes);
V(:,1) = p0;

% Fill Dynamic Programming table
for t = 2:nNodes
    [V(1,t),T(1,t)] = max([V(1,t-1)*pT(1,1,t-1) V(2,t-1)*pT(2,1,t-1)]);
    [V(2,t),T(2,t)] = max([V(1,t-1)*pT(1,2,t-1) V(2,t-1)*pT(2,2,t-1)]);
end

% Work backwords to get optimal solution path
yOpt = zeros(nNodes,1);
[~,yOpt(end)] = max(V(:,end));
for n = nNodes:-1:2
   yOpt(n-1) = T(yOpt(n),n);
end
```

The optimal decoding is 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 2 2 1 2 1 1 2 2 2 1 1 1 1 1 1 2. Notice that this is quite different from the sequence maximizing the marginals.

1.2 Conditioning

The long sequence from the previous question usually starts with state 1 and most of the time ends in state 2. In this question you'll consider conditioning on these events not happening. First, compute the following quantities which can be done using your functions from the previous question:

1. Report all the univariate conditional probabilities $p(x_j|x_1 = 2)$ obtained using a Monte Carlo estimate based on 10000 samples.
2. Report all the exact univariate conditionals $p(x_j|x_1 = 2)$.
3. Report the sequence beginning with $x_1 = 2$ that has the highest probability.
4. Report the sequence ending with $x_d = 1$ that has the highest probability.

Hint: these conditions can be done by changing the input to the functions from the previous question.

Next consider the following cases (which require implementing an extra rejection step or backward phase):

5. Report all the univariate conditional probabilities $p(x_j|x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples and rejection sampling. Also report the number of samples accepted among the 10000 samples.
6. Write a function, *sampleBackwards.m* that uses backwards sampling to sample sequences where $x_d = 1$). Hand in this code and report all the univariate conditional probabilities $p(x_j|x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples.
7. Write a function, *forwardBackwards.m* that is able compute all exact univariate conditionals $p(x_j|x_d = 1)$ in $O(dk^2)$. Hand in the code and report all the exact univariate conditionals $p(x_j|x_d = 1)$.

Answer:
For the first four parts, you can set $p0 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ and then call the codes from the previous question:

1. This gives the marginal probabilities for state 1 as 0 0.0611 0.9294 0.9752 0.7841 0.2201 0.0531 0.3350 0.3236 0.5887 0.3591 0.7472 0.3740 0.1283 0.3605 0.4005 0.7215 0.7888 0.3107 0.6726 0.5795 0.2554 0.4021 0.2364 0.7425 0.6250 0.6058 0.7361 0.4679 0.4444 0.1767.
2. The marginals for state 1 are given by 0 0.0634 0.9297 0.9756 0.7789 0.2265 0.0542 0.3359 0.3204 0.5902 0.3601 0.7479 0.3726 0.1313 0.3744 0.4103 0.7240 0.7877 0.3147 0.6698 0.5811 0.2541 0.4037

0.2403 0.7378 0.6211 0.6060 0.7419 0.4644 0.4380 0.1782.

3. The optimal decoding in this setting is the same as the optimal decoding, except the first two states are 2. Indeed, from the Markov property we should expect it to be the same as the optimal decoding starting from the first state where they agree.

4. The decoding under this condition is the same as the optimal decoding, except that we end with 1.

5. On one run I got 1820 samples accepted (about 18% which agrees with the marginal probability) and got the following marginal for state 1: 0.5775 0.4495 0.8989 0.9753 0.7676 0.2187 0.0484 0.3242 0.2973 0.5890 0.3511 0.7495 0.3901 0.1363 0.3747 0.4060 0.7335 0.8000 0.3049 0.6588 0.5813 0.2407 0.4137 0.2412 0.7418 0.6203 0.6033 0.7253 0.4291 0.3280 1.0000.

6. The code could look like this:

```
function [X] = backwardSample(p0,pT,nSamples,xd)
nNodes = size(pT,3)+1;

V = zeros(2,nNodes);
% Fill Dynamic Programming table
V(:,1) = p0;
[~,yOpt(1)] = max(V(:,1));
for t = 2:nNodes
    V(:,t) = pT(:,:,t-1)'*V(:,t-1);
end

X = zeros(nSamples,nNodes);
for i = 1:nSamples
    X(i,nNodes) = xd;
    for j = nNodes-1:-1:1
        pT_backwards = pT(:,X(i,j+1),j).*V(:,j);
        pT_backwards = pT_backwards/sum(pT_backwards);
        if rand < pT_backwards(1)
            X(i,j) = 1;
        else
            X(i,j) = 2;
        end
    end
end
```

and the marginals from one run for me were 0.5912 0.4588 0.8999 0.9741 0.7751 0.2246 0.0529 0.3389 0.3232 0.5909 0.3590 0.7515 0.3673 0.1351 0.3753 0.4125 0.7228 0.7870 0.3098 0.6695 0.5870 0.2520 0.3990 0.2358 0.7439 0.6169 0.6000 0.7304 0.4281 0.3284 1.0000.

7. The code could look like this:

```
function [marginals] = forwardBackward(p0,pT,xd)
nNodes = size(pT,3)+1;

V = zeros(2,nNodes);
% Forward phase
V(:,1) = p0;
[~,yOpt(1)] = max(V(:,1));
for t = 2:nNodes
    V(:,t) = pT(:,:,t-1)'*V(:,t-1);
end

% Backward phase if final state is xd
B = zeros(2,nNodes);
B(xd,nNodes) = 1;
for j = nNodes-1:-1:1
    B(:,j) = pT(:,:,j)*B(:,j+1);
end

marginals = V.*B; % Marginals are proportional to these
marginals = marginals*diag(1./sum(marginals)); % Normalize columns
```

and the exact marginals are given by 0.6000 0.4681 0.8998 0.9718 0.7779 0.2267 0.0542 0.3359 0.3204 0.5902 0.3601 0.7479 0.3726 0.1313 0.3744 0.4103 0.7240 0.7877 0.3147 0.6698 0.5811 0.2541 0.4037 0.2403 0.7378 0.6211 0.6059 0.7324 0.4318 0.3296 1.0000. Notice that the early marginals in the sequence are almost identical to the unconditional case, while the later ones are different.

1.3 1D Linear-Gaussian Markov Chains

Consider a continuous-state Markov chain where the initial distribution is given by

$$x_0 \sim \mathcal{N}(m_0, v_0^2),$$

and the transition distributions for $j > 1$ are given by

$$x_j | x_{j-1} \sim \mathcal{N}(w_j x_{j-1} + m_j, v_j^2).$$

This model could be used to model an object moving through $\mathbb{R}$.[1] Because of the Gaussian assumptions, this defines a joint Gaussian distribution over the variables while the marginal distributions are also Gaussian. For a generic $j > 1$, derive the form of the marginal distribution of x_j, expressing the marginal parameters μ_j and σ_j recursively in terms of μ_{j-1} and σ_{j-1}.

Hint: You can use Theorem 4.4.1 of Murphy's book. Answer:

We have

$$\begin{aligned} p(x_j) &= \int_{x_{j-1}} p(x_j, x_{j-1}) dx_{j-1} \\ &= \int_{x_{j-1}} p(x_j | x_{j-1}) p(x_{j-1}) dx_{j-1}. \end{aligned}$$

so the result is the normalizing constant of joint Gaussian defined by

$$p(x_{j-1}) = \mathcal{N}(\mu_{j-1}, \sigma_{j-1}^2), \quad p(x_j | x_{j-1}) = \mathcal{N}(w_j x_{j-1} + m_j, v_j^2).$$

This is a special case of Theorem 4.4.1 with the following replacements:

$$\begin{aligned} x &\leftarrow x_{j-1} \\ y &\leftarrow x_j \\ \mu_x &\leftarrow \mu_{j-1} \\ \Sigma_x &\leftarrow \sigma_{j-1}^2 \\ A &\leftarrow w_j \\ b &\leftarrow m_j \\ \Sigma_y &\leftarrow \sigma_j^2. \end{aligned}$$

Plugging these values into the normalizing constant result $p(y)$ we get

$$x_j \sim \mathcal{N}(w_j \mu_{j-1} + m_j, \sigma_j^2 + w_j^2 \sigma_{j-1}^2).$$

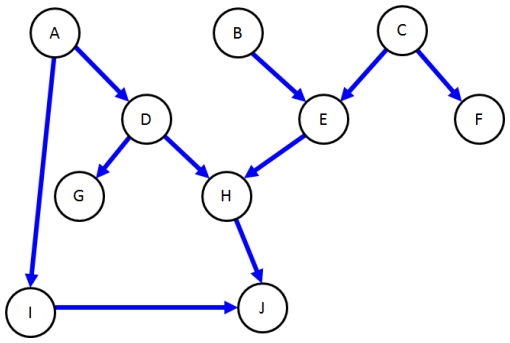
The mean makes sense as it's the previous mean transformed by w_j and m_j. We see that the variance is given by the variance of the conditional added to the variance of the previous marginal multiplied by w^2.

2 Directed Acyclic Graphical Models

2.1 D-Separation

Consider a directed acyclic graphical (DAG) model with the following graph structure:

[1] In practical applications like object tracking, we typically have that the states x_j are 2- or 3-dimensions so we model an object like a submarine or an airplane moving through space.



Assuming that the conditional independence properties are faithful to the graph, using d-separation briefly explain why the following are true or false:

1. $B \perp F$.
2. $B \perp F \mid A$.
3. $B \perp F \mid C$.
4. $B \perp F \mid E$.
5. $B \perp F \mid I$.
6. $B \perp F \mid J$.
7. $B \perp F \mid C, E$.

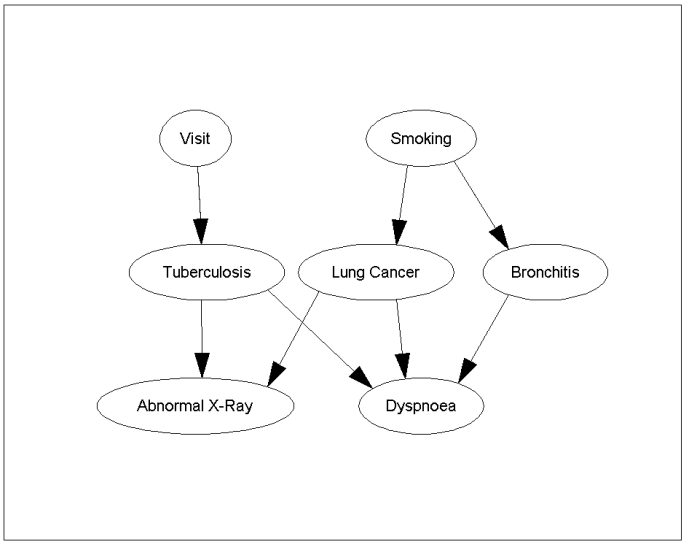
Answer:

From the graph, B and F will be d-separated if we see C, or if we don't see E or any of its descendents. It follows that we have:

1. True (not seeing E blocks the path).
2. True (A is not a descendent of E, so doesn't unblock the path).
3. True (seeing C doubly-blocks the path).
4. False (seeing E removes the d-separation and creates a path).
5. True (I is no a descendent of E).
6. False (J is a descendent of E so unblocks the path).
7. True (seeing C blocks the path).

2.2 Exact Inference

While DAGs can be used as a visual representation of independence assumptions, they can also be used to simplify computations.This question will give you practice using the basic properties which allow efficient computations in graphical models. Consider the DAG model below, for distinguishing between different causes of shortness-of-breath (dyspnoea) and the causes of an abnormal lung x-ray, while modelling potential causes of these diseases too (whether the person is a smoker or had a 'visit' to a country with a high degree of tuberculosis).



For this question, let's assume that we use the following parameterization of the network:

$$\text{Visit}$$
$$p(V = 1) = 0.01$$
$$\text{Smoking}$$
$$p(S = 1) = 0.2$$
$$\text{Tuberculosis}$$
$$p(T = 1|V = 1) = 0.05$$
$$p(T = 1|V = 0) = 0.01$$
$$\text{Lung Cancer}$$
$$p(L = 1|S = 1) = 0.10$$
$$p(L = 1|S = 0) = 0.01$$
$$\text{Bronchitis}$$
$$p(B = 1|S = 1) = 0.60$$
$$p(B = 1|S = 0) = 0.30$$
$$\text{Abnormal X-Ray}$$
$$p(X = 1|T = 1, L = 1) = 1.00$$
$$p(X = 1|T = 1, L = 0) = 0.98$$
$$p(X = 1|T = 0, L = 1) = 0.9$$
$$p(X = 1|T = 0, L = 0) = 0.05$$

Dyspnoea

$$p(D=1|T=1,L=1,B=1)=0.90$$
$$p(D=1|T=1,L=1,B=0)=0.70$$
$$p(D=1|T=1,L=0,B=1)=0.85$$
$$p(D=1|T=1,L=0,B=0)=0.65$$
$$p(D=1|T=0,L=1,B=1)=0.82$$
$$p(D=1|T=0,L=1,B=0)=0.60$$
$$p(D=1|T=0,L=0,B=1)=0.80$$
$$p(D=1|T=0,L=0,B=0)=0.10$$

Compute the following quantities (hints are given on the right, and these will be easier to do in order and if you use conditional independence properties to simplify the calculations):

0. $p(S=1)$ (marginal of root node; can read from table)
1. $p(S=0)$ (negation of marginal of root node; use sum to one constraint)
2. $p(L=1|S=1)$ (conditional of child node given parents; can be read from table)
3. $p(L=1)$ (marginal of child node; marginalize over parent)
4. $p(X=1|T=1,L=1)$ (conditional of child given parents; can be read from table)
5. $p(X=1|T=1)$ (conditional of child with missing parent; marginalize over missing parent)
6. $p(X=1|T=1,S=1)$ (conditional of child given parent and grand-parent, marginalize over missing parent)
7. $p(X=1)$ (marginal of leaf node; marginalize over parents and use independence to simplify)
8. $p(T=1|X=1)$ (conditional of parent given child; use Bayes rule)
9. $p(T=1|L=1)$ (conditional of parent given co-parent; use independence and then marginal)
10. $p(T=1|X=1,L=1)$ (conditional of parent given child and co-parent; use Bayes rule)

Answers:

0. From the table,
$$p(S=1)=0.20.$$

1. From the table and sum-to-one constraint,
$$p(S=0)=1-p(S=1)=0.80.$$

2. From the table,
$$p(L=1|S=1)=0.10.$$

3. Marginalizing over the parent S we get
$$\begin{aligned}p(L=1)&=p(L=1,S=1)+p(L=1,S=0)\\&=p(L=1|S=1)p(S=1)+p(L=1|S=0)p(S=0)\\&=(0.10)(0.2)+(0.01)(0.8)\\&=0.028\end{aligned}$$

4. From the table,

$$p(X = 1|T = 1, L = 1) = 1.00.$$

5. Marginalizing over the missing parent L and using independence between L and T we have

$$\begin{aligned}
&p(X = 1|T = 1)\\
&= p(X = 1, L = 1|T = 1) + p(X = 1, L = 0|T = 1)\\
&= p(X = 1|T = 1, L = 1)p(L = 1|T = 1) + p(X = 1|T = 1, L = 0)p(L = 0|T = 1)\\
&= p(X = 1|T = 1, L = 1)p(L = 1) + p(X = 1|T = 1, L = 0)p(L = 0)\\
&= (1.00)(0.028) + (0.98)(1 - 0.028)\\
&0.9806
\end{aligned}$$

This is lower than if we knew they were a smoker.

6. Marginalizing over the missing parent L and using independence between L and T

$$\begin{aligned}
&p(X = 1|T = 1, S = 1)\\
&= p(X = 1, L = 1|T = 1, S = 1) + p(X = 1, L = 0|T = 1, S = 1)\\
&= p(X = 1|T = 1, L = 1, S = 1)p(L = 1|T = 1, S = 1) + p(X = 1|T = 1, L = 0, S = 1)p(L = 0|T = 1, S = 1)\\
&= p(X = 1|T = 1, L = 1)p(L = 1|S = 1) + p(X = 1|T = 1, L = 0)p(L = 0|S = 1)\\
&= (1.00)(0.10) + (0.98)(1 - 0.10)\\
&= 0.9820
\end{aligned}$$

This is lower then the 1.00 we get if we explicitly know they have lung cancer.

7. To get this quantity, we need to have the marginals of T and L. We computed it with respect to L above and the marginal with respect to T we get by marginalizing over V,

$$\begin{aligned}
p(T = 1) &= p(T = 1, V = 0) + p(T = 1, V = 1)\\
&= p(T = 1|V = 0)p(V = 0) + p(T = 1|V = 1)p(V = 1)\\
&= (0.01)(1 - .01) + (0.05)(0.01)\\
&= 0.0104.
\end{aligned}$$

Observe that this is smaller than $p(T = 1|V = 1) = 0.05$ and larger than $p(T = 1|V = 0) = 0.01$.

Now marginalizing over the missing parents T and L and using independence between T and L to simplify we get

$$\begin{aligned}
p(X = 1) &= \sum_{t,l} p(X = 1, T = t, L = l)\\
&= \sum_{t,l} p(X = 1|T = t, L = l)p(T = t, L = l)\\
&= \sum_{t,l} p(X = 1|T = t, L = l)p(T = t)p(L = l)\\
&= (1.00)(0.0104)(0.028) + (0.98)(0.0104)(1 - 0.028) + (0.9)(1 - 0.0104)(0.028) + (0.05)(1 - 0.0104)(1 - 0.028)\\
&= 0.0832
\end{aligned}$$

This is a very simple instance of belief propagation on a tree. The 'messages' sent to to X are the marginals $p(T = t)$ and $p(L = l)$, which (by their independence) summarize what we need about the joint distribution of T and L.

8. Using Bayes rule so that we can use the known distribution of the child given the parent, $p(X = 1|T = 1)$, that we computed above,

$$\begin{aligned} p(T = 1|X = 1) &= \frac{p(X = 1|T = 1)p(T = 1)}{p(X = 1)} \\ &= (0.9806)(0.0104)/0.0832 \\ &= 0.1226 \end{aligned}$$

This is about ten higher than the prior probability, $p(T = 1)$, before we knew the x-ray was abnormal, but it is still unlikely because the abnormal x-ray could be explained by lung cancer rather than TB or (more likely) it could be explained by the 0.05 probability of having an abnormal x-ray even without TB or lung cancer.

9. The co-parents are independent if we don't know their child, so use independence and the marginal we computed above.

$$p(T = 1|L = 1) = p(T = 1) = 0.0104.$$

Knowing lung cancer on its own doesn't tell us anything about TB.

10. To compute this quantity, we'll need the conditional of X given L, which we get by marginalizing over the missing parent T,

$$\begin{aligned} &p(X = 1|L = 1) \\ &= p(X = 1, T = 1|L = 1) + p(X = 1, T = 0|L = 1) \\ &= p(X = 1|T, = 1, L = 1)p(T = 1|L = 1) + p(X = 1|T = 0, L = 1)p(T = 0|L = 1) \\ &= p(X = 1|T, = 1, L = 1)p(T = 1) + p(X = 1|T = 0, L = 1)p(T = 0) \\ &= (1.00)(0.0104) + (0.9)(1 - .0104) \\ &= 0.901 \end{aligned}$$

Now use Bayes rule to get

$$\begin{aligned} p(T = 1|X = 1, L = 1) &= \frac{p(X = 1|L = 1, T = 1)p(T = 1|L = 1)}{p(X = 1|L = 1)} \\ &= \frac{p(X = 1|T = 1, L = 1)p(T = 1)}{p(X = 1|L = 1)} \\ &= \frac{(1.00)(0.0104)}{0.901} \\ &= 0.0115 \end{aligned}$$

Knowing that they have lung cancer significantly decreases the probability that the abnormal x-ray was due to TB, to the point that it's only slightly higher than the marginal probability, $p(T = 1)$. This is 'explaining away' between the two causes, TB and lung cancer, of the abnormal x-ray.

2.3 Inpainting

The function *example_fil.m* loads a variant of the MNIST dataset. It contains all the training images but the test images are missing their bottom half. Running this function fits an independent Bernoulli model to the training set, and then shows the result of applying the density model to "fill in" four random test examples. It performs pretty badly because the independent model can't condition on the known top-half of the images.

1. Make a variant of the demo where you fit an inhomogeneous Markov chain to each image column. Hand in your code and an example of using this model to fill in 4 random test images.

2. Make a variant of the demo where you fit a directed acyclic graphical model to the data, using general discrete conditional probabilities and where the parents of pixel (i, j) are the other 8 pixels in the region $(i-2:i, j-2:j)$. Hand in your code and an example of using this model to fill in 4 random test images.

3. Consider using more than 8 pixels are parents in the above model, such as the 15 pixels in the region $(i-3:i, j-3:j)$. If you do this, the code will often place white pixels in the bottom right corner of the image even though no training example has a white pixel there. Why would it do this?

4. Make a variant of the demo where you fit a sigmoid belief network to the data, where the parents of pixel (i, j) are the other pixels in the region $(1:i, 1:j)$. Hand in your code and an example of using this model to fill in 4 random test images.

Hints: For parts 2-4, you may find it helpful to make an m by m cell array called *models* where element (i, j) contains the model for pixel (i, j). For these parts the size of the dataset also mean you will probably need to vectorize your computation. The functions *permute* and *reshape* will help you, making a sparse version of X with *sparse* can also speed up many operations. For part 2, you can use *binaryTabular.m* to fit the discrete conditional distribution and sample from it (a reasonable value of α is 1). For part 4, you can use *logisticL2.m* to fit logistic regression models and sample from them (a reasonable value of λ is 1). Note that *logisticL2.m* uses a $\{-1, 1\}$ encoding of y while *binaryTabular.m* uses a $\{0, 1\}$ encoding (both support sparse X).

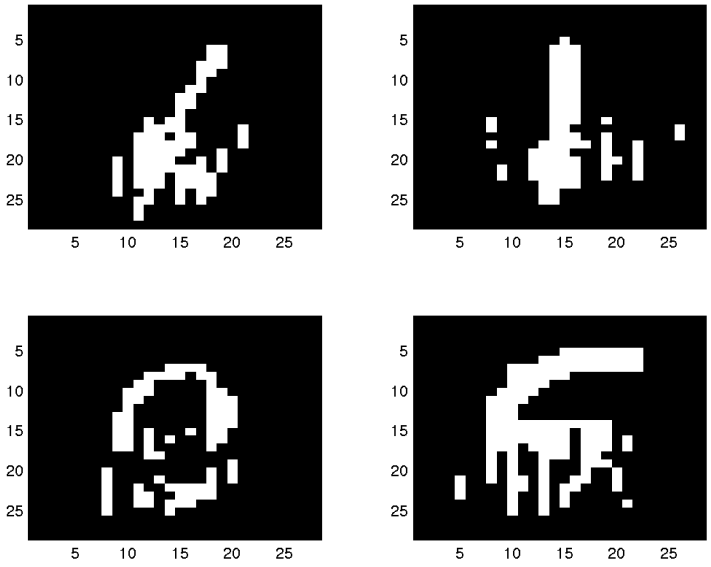
Answer:

1. The Markov chain could look like this

```
% Initial probabilities
for j = 1:m
    p_ijk(1,j) = sum(X(1,j,:) == 1)/n;
end

% Transition probabilities
for i = 2:m
    for j = 1:m
        p_ijk(i,j,1) = sum(X(i-1,j,:) == 1 & X(i,j,:) == 1)/sum(X(i-1,j,:) == 1);
        p_ijk(i,j,2) = sum(X(i-1,j,:) == 0 & X(i,j,:) == 1)/sum(X(i-1,j,:) == 0);
    end
end
```

   Some images filled-in with this model look like:



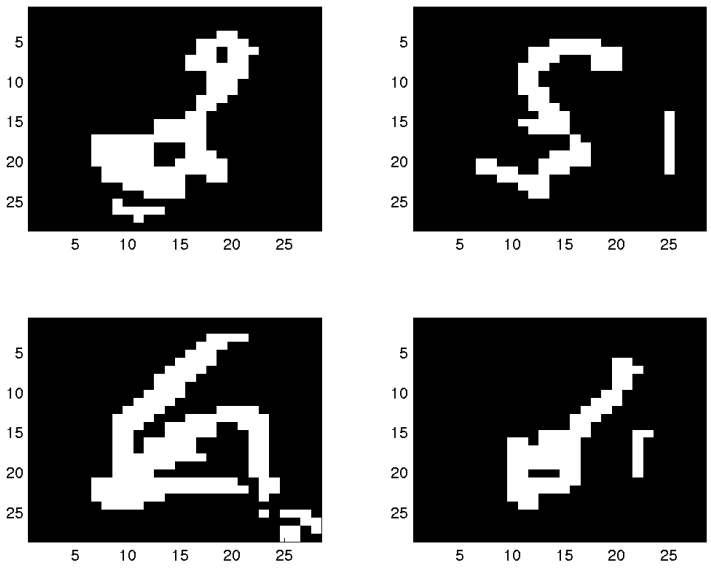
2. The tabular DAG could look like this

```
models = cell(m,m);
Xperm = permute(X,[3 1 2]);
offset = 2;
for i = 1:m
    i
    for j = 1:m
        % Make features (needs to only consider small up/left part)
        Xsub = Xperm(:,max(1,i-offset):i,max(1,j-offset):j);
        Xsub = reshape(Xsub,[n numel(Xsub)/n]);
        Xsub = Xsub(:,1:end-1);
        Xsub = sparse(Xsub);

        % Make labels
        y = X(i,j,:);
        y = y(:);

        % Fit discrete distribution
        models{i,j} = binaryTabular(Xsub,y,1);
    end
end
```

Some images filled-in with this model look like:



3. The problem is that there are huge number of possible parents 2^15, and due to the random sampling you end up with some parent configurations that are not in the training data. In these case the model relies on the Laplace smoothing to do everything, but this says that both 0 and 1 are equally likely. A better model would probably apply Laplace smoothing in a more clever way.

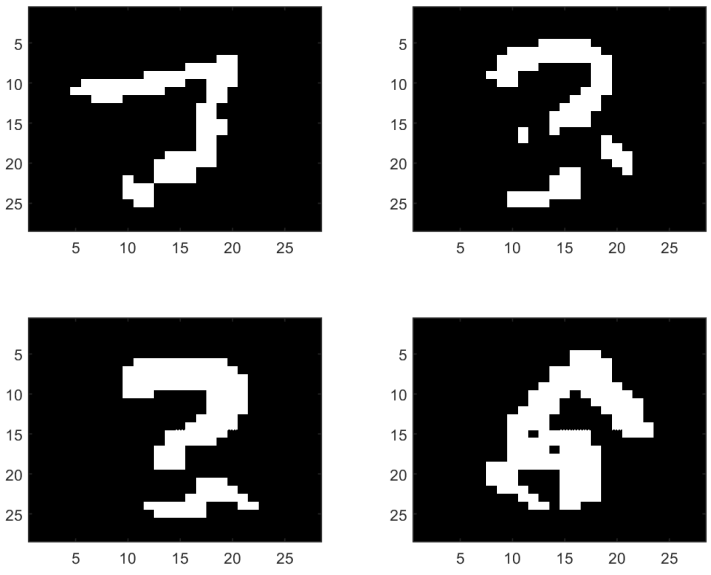
4. The sigmoid DAG could look like this

```
models = cell(m,m);
Xperm = permute(X,[3 1 2]);
for i = 1:m
    i
    for j = 1:m
        % Make features
        Xsub = Xperm(:,1:i,1:j);
        Xsub = reshape(Xsub,[n i*j]);
        Xsub = Xsub(:,1:end-1);
        Xsub = sparse(Xsub);

        % Make labels
        y = X(i,j,:);
        y = y(:);
        y(y == 0) = -1;

        % Fit logistic regression model
        models{i,j} = logisticL2(Xsub,y,1);
    end
end
```

Some images filled-in with this model look like:



3 Paper Review

3.1 Finding Relevant Papers

To help you make progress on your project, for this part you should hand in a list of 10 academic papers related to your current project topic. Finding related work is often one of the first steps towards getting a new project started, and it gives you an idea of what has (and has not) been explored. Some strategies for finding related papers are:

1. Use Google: try the keywords you think are most relevant. Asking people in your lab (or related labs) for references is also often a good starting point.
2. Once you have found a few related papers, read their introduction section to find references that these papers think are worth mentioning.
3. Once you have found a few related papers, use Google Scholar to look through the list of references that are *citing* these papers. You may have to do some sifting if there are a lot of citations. Reasonable criteria to sift through large reference lists include looking for the ones with the most citations or focusing on the more recent ones (then returning to Step 2 to find the more-relevant older references).

For this question you only need to provide a list, but in Assignment 5 you will have to do a survey of 10 papers. So it's worth trying to identify papers that are both relevant and important at this point. For some types of projects it will be easier to find papers than others. If you are having trouble, post on Piazza.

Although the papers do not need to all be machine learning papers, the course project does need to be related to machine learning in some way, so at least a subset of the papers should be machine learning papers. Here is a rough guide to some of the most reputable places to where you see machine learning works published:

- The International Conference on Machine Learning (ICML) and the conference on Advances in Neural Information Processing (NIPS) are the top places to publish machine learning work. The Journal of Machine Learning Research (JMLR) is the top journal, although in this field conference publications are usually viewed as more prestigious.
- Other good venues include AISTATS (emphasis on statistics), UAI (emphasis on graphical models), COLT (emphasis on theory), ICLR (emphasis on deep learning), ECML-PKDD (European version of ICML), CVPR and ICCV/ECCV (emphasis on computer vision), ACL and EMNLP (emphasis on language), KDD (emphasis on data mining), AAAI/IJCAI (emphasis on AI more broadly), JRSSB and Annals of Stats (emphasis on statistics more broadly), and Science and Nature (emphasis on science more broadly).

3.2 Paper Review

Among your list of 10 papers, choose one paper and write a review of this paper. It makes sense to choose one of the most closely-related to your project or one of the most important-looking papers. The review should have two parts:

1. A short summary of the contributions of the paper. Say what problem the paper is addressing, why this is an important problems, what is proposed, and how it is being evaluated.
2. A list of strengths and weaknesses of the paper. For ideas of what issues to discuss, see the JMLR guidelines for reviewers: `http://www.jmlr.org/reviewer-guide.html`

Note that you should include a non-empty list of strengths *and* weaknesses. Many students when doing their first reviews focus either purely on strengths or purely on weaknesses. It's important to recognize that all papers have weaknesses or limitations (even ones written by famous people or that are published

in impressive places or that proved to be historically important) and all papers have strengths or at least motivation (the authors must have thought it was worth writing for some reason).