CPSC 540 Assignment 4 (due March 14 at midnight)

The assignment instructions are the same as for the previous assignment.

1. Name(s):
2. Student ID(s):

1 Discrete Markov Chains

1.1 Sampling, Inference, and Decoding

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

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

It then finds the optimal decoding (the most likely assignment to the variables $\{x_1, x_2, \dots, x_d\}$) in the first two 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 to decode the last chain because it has 31 variables. In this question you'll explore two ways to estimate the marginals in the third Markov chain and two ways to estimate the most-probable sequence (you only need to report results on the third Markov chain for this question, but the first two may be helpful for debugging).

1. Write a function, *sampleAncestral*, that uses ancestral sampling to sample a sequence x. Hand in this code and report all the univariate marginal probabilities using a Monte Carlo estimate based on 10000 samples.

   Answer: The code could look roughly like this:

```
function sampleAncestral(p0,pT,t)
    d = size(pT,3)+1

    X = ones(Int64,t,d)
    for i in 1:t
        if rand() > p0[1]
            X[i,1] = 2
        end
        for j in 2:d
            if rand() > pT[X[i,j-1],1,j-1]
                X[i,j] = 2
            end
        end
    end
    return X
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. Write a function, *marginalCK*, that uses the CK equations to compute the exact univariate marginals. Hand in this code, report all exact univariate marginals, and report how this differs from the marginals in the previous question.

   Answer: The code could look roughly like this:

```
function marginalCK(p0,pT)
    d = size(pT,3)+1

    M = zeros(2,d)
    M[:,1] = p0
    for j in 2:d
        M[:,j] = pT[:,:,j-1]'*M[:,j-1]
    end
    return M
end
```

   The exact marginals for the first state 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. Write a function, *viterbiDecode*, that implements the Viterbi decoding algorithm for Markov chains. Hand in this code and report the optimal decoding of the third Markov chain, and how this differs from the sequence defined by the most likely state at each time (the states maximizing the marginal probabilities).

   Answer: The code should look roughly like this:

```
function viterbiDecode(p0,pT)
    d = size(pT,3)+1

    # Fill out dynamic programming table
    B = zeros(2,d)
    M = zeros(2,d)
    M[:,1] = p0
    for j in 2:d
        (M[1,j],B[1,j]) = findmax([M[1,j-1]*pT[1,1,j-1] M[2,j-1]*pT[2,1,j-1]])
        (M[2,j],B[2,j]) = findmax([M[1,j-1]*pT[1,2,j-1] M[2,j-1]*pT[2,2,j-1]])

    end

    # Work backwards to get an optimal solution
    decoding = zeros(Int64,d)
    (~,decoding[end]) = findmax(M[:,end])
    for j in d:-1:2
        decoding[j-1] = B[decoding[j],j]
    end
    return decoding
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 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.

Hint: for parts 2-3, you can use a 2 by d matrix M to represent the dynamic programming table, and for part 3 you can use another matrix B containing the argmax values that lead to each entry in the table.

1.2 Simple Conditioniing Queries

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 $p(x_j \mid x_1 = 2)$ obtained using a Monte Carlo estimate based on 10000 samples.

   Answer: you can set $p0 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ and then call the codes from the previous question. 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. Report all the exact univariate conditionals $p(x_j \mid x_1 = 2)$.

   Answer: 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. Report the sequence beginning with $x_1 = 2$ that has the highest probability. How does this differ from the (unconditional) optimal decoding?

   Answer: 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. Report the sequence ending with $x_d = 1$ that has the highest probability. How does this differ from the (unconditional) optimal decoding?

   Answer: You can obtain by either (i) changing the probability of transitioning to 2 in the last state to be zero, (ii) using by modifying the first step of the the backtracking part of the Viterbi code, (iii) modifying the message M in the last part of the forward pass of the Viterbi code, or (iv) tranposing all the transition probabilities and changing the initial transition probabilities (as well as the final ones) appropriately). The decoding under this condition is the same as the optimal decoding, except that we end with 1.

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

1.3 Conditioning Queries Requiring Inference

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

1. Report all the univariate conditional probabilities $p(x_j \mid x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples and rejection sampling. Also report the number of samples accepted among the 10000 samples.

   Answer: 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.

2. Write a function, *sampleBackwards* that uses backwards sampling to sample sequences where $x_d = 1$). Hand in this code and report all the univariate conditional probabilities $p(x_j \mid x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples.

   Answer: The code could look like this:

```
function backwardSample(p0,pT,t,xd)
    d = size(pT,3)+1

    M = marginalCK(p0,pT)

    X = ones(Int64,t,d)
    for i in 1:t
        X[i,d] = xd
        for j in d-1:-1:1
            pT_back = pT[:,X[i,j+1],j].*M[:,j]
            pT_back ./= sum(pT_back)
            if rand() > pT_back[1]
                X[i,j] = 2
            end
        end
    end
    return X
```

   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.

3. Write a function, *forwardBackwards* that is able compute all exact univariate conditionals $p(x_j \mid x_d = 1)$ in $O(dk^2)$. Hand in the code and report all the exact univariate conditionals $p(x_j \mid x_d = 1)$.

   Answer: The code could look like this:

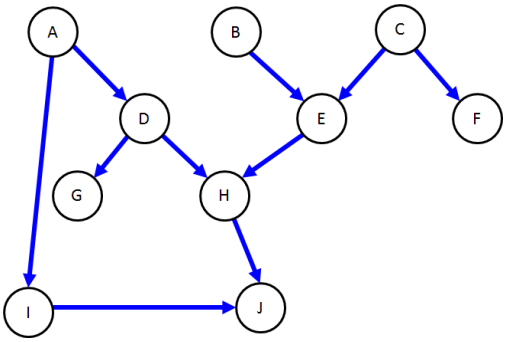
```
function forwardBackward(p0,pT,xd)
    d = size(pT,3)+1

    M = marginalCK(p0,pT)

    V  = zeros(2,d)
    V[xd,d] = xd
    for j in d-1:-1:1
        V[:,j] = pT[:,:,j]*V[:,j+1]
    end
    marginals = M.*V
    for j in 1:d
        marginals[:,j] ./= sum(marginals[:,j])
    end
    return marginals
end
```

   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.

2 Directed Acyclic Graphical Models

2.1 D-Separation

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



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

1. $B \perp F$.

   Answer: True (not seeing E blocks the path).

2. $B \perp F \mid A$.

   Answer: True (A is not a descendent of E, so doesn't unblock the path).

3. $B \perp F \mid C$.

   Answer: True (seeing C doubly-blocks the path).

4. $B \perp F \mid E$.

   Answer: False (seeing E removes the d-separation and creates a path).

5. $B \perp F \mid I$.

   Answer: True (I is no a descendent of E).

6. $B \perp F \mid J$.

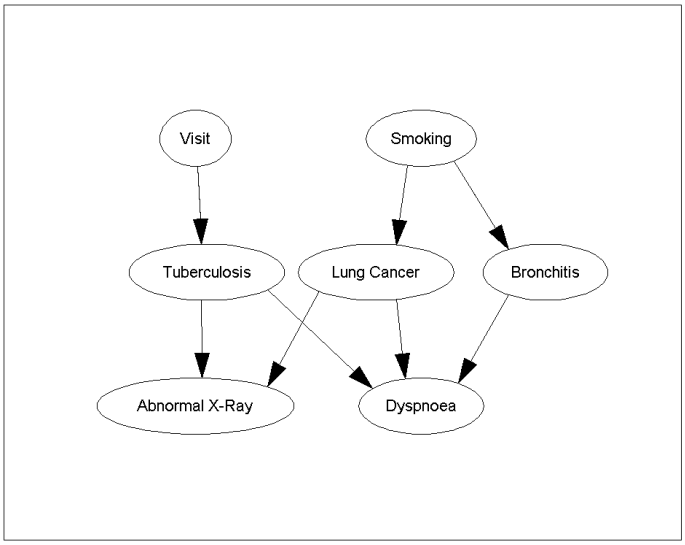
   Answer: False (J is a descendent of E so unblocks the path).

7. $B \perp F \mid C, E$.

   Answer: 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:

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

Dyspnoea

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

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):

1. $p(S = 0)$ (negation of marginal of root node; use sum to one constraint).

   Answer: From the table and sum-to-one constraint,

   $$p(S = 0) = 1 - p(S = 1) = 0.80.$$

2. $p(L = 1)$ (marginal of child node; marginalize over parent).

   Answer: Marginalizing over the parent S we get

   $$\begin{aligned} p(L = 1) &= p(L = 1, S = 1) + p(L = 1, S = 0) \\ &= p(L = 1|S = 1)p(S = 1) + p(L = 1|S = 0)p(S = 0) \\ &= (0.10)(0.2) + (0.01)(0.8) \\ &= 0.028 \end{aligned}$$

3. $p(X = 1 \mid T = 1)$ (conditional of child with missing parent; marginalize over missing parent).

   Answer: 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.

4. $p(X = 1 \mid T = 1, S = 1)$ (conditional of child given parent and grand-parent, marginalize over missing parent).

Answer: 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.

5. $p(X=1)$ (marginal of leaf node; marginalize over parents and use independence to simplify).

   Answer: 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.

6. $p(T=1 \mid X=1)$ (conditional of parent given child; use Bayes rule).

   Answer: 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.

7. $p(T = 1 \mid L = 1)$ (conditional of parent given co-parent; use independence and then marginal).

   Answer: 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.

8. $p(T = 1 \mid X = 1, L = 1)$ (conditional of parent given child and co-parent; use Bayes rule).

   Answer: 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

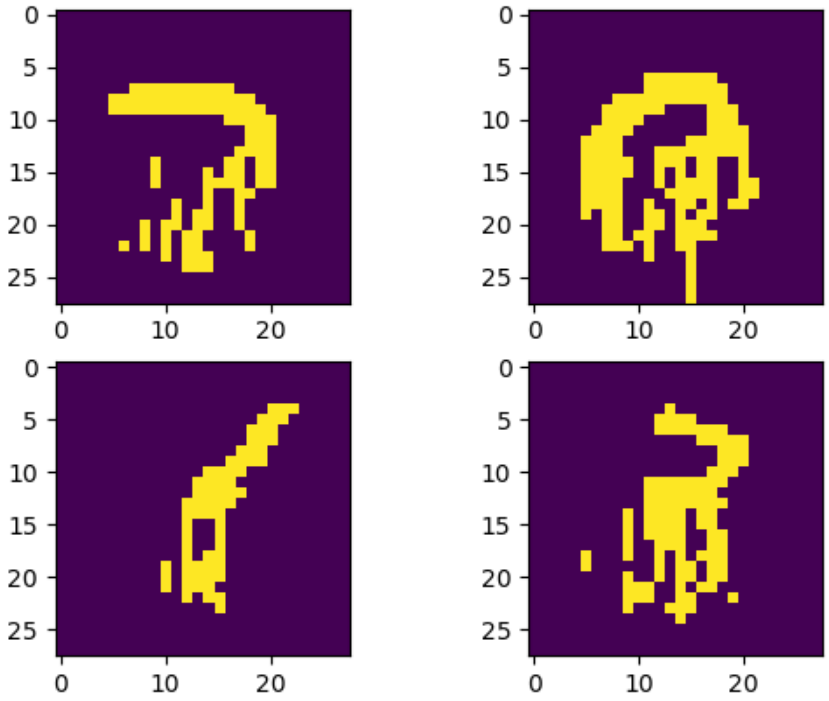
The function *example_fil.jl* loads a variant of the MNIST dataset. It contains all the training images but the test images are missing their bottom half. Running this function fits an independent Bernoulli model to the training set, and then shows the result of applying the density model to "fill in" 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.

   Answer: The Markov chain could look like this

```
# Initial probabilities
p_ijk = zeros(m,m,2)
for j in 1:m
    p_ijk[1,j,1] = sum(X[1,j,:] .== 1)/n
end

# Transition probabilities
for i in 2:m
    for j in 1:m
        c1 = sum(X[i-1,j,:] .== 1)
        p_ijk[i,j,1] = sum((X[i-1,j,:] .== 1) .& (X[i,j,:] .== 1))/c1
        p_ijk[i,j,2] = sum((X[i-1,j,:] .== 0) .& (X[i,j,:] .== 1))/(n-c1)
    end
end
```

Some images filled-in with this model look like:

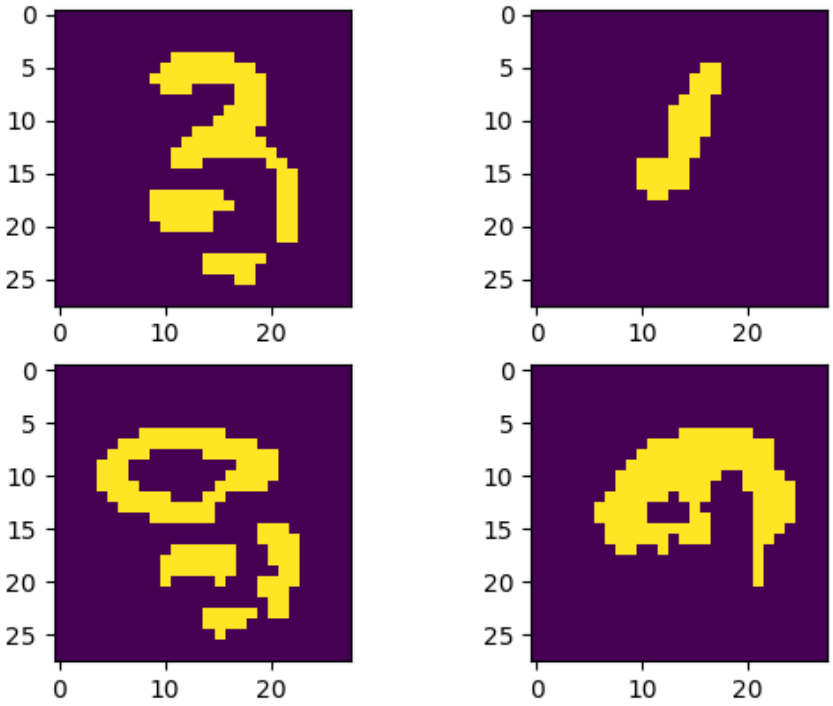


2. Make a variant of the demo where you fit a directed acyclic graphical model to the data, using general discrete conditional probabilities and where the parents of pixel (i, j) are the other 8 pixels in the region $(i-2:i, j-2:j)$. Hand in your code and an example of using this model to fill in 4 random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

   Answer: The tabular DAG could look like this

```
models = Array{Any}(m,m)
include("tabular.jl")
@printf("Fitting...\n")
offset = 2
for i in 15:m
    for j in 1:m
       @show (i,j)
        i1 = max(i-offset,1)
        j1 = max(j-offset,1)

        d = (i-i1+1)*(j-j1+1)
        Xsub = zeros(n,d)
        k = 1
        for ii in i1:i
            for jj in j1:j
                Xsub[:,k] = X[ii,jj,:]
                k += 1
            end
        end
        models[i,j] = tabular(Xsub[:,1:d-1],Xsub[:,d])
    end
end
```

Some images filled-in with this model look like:

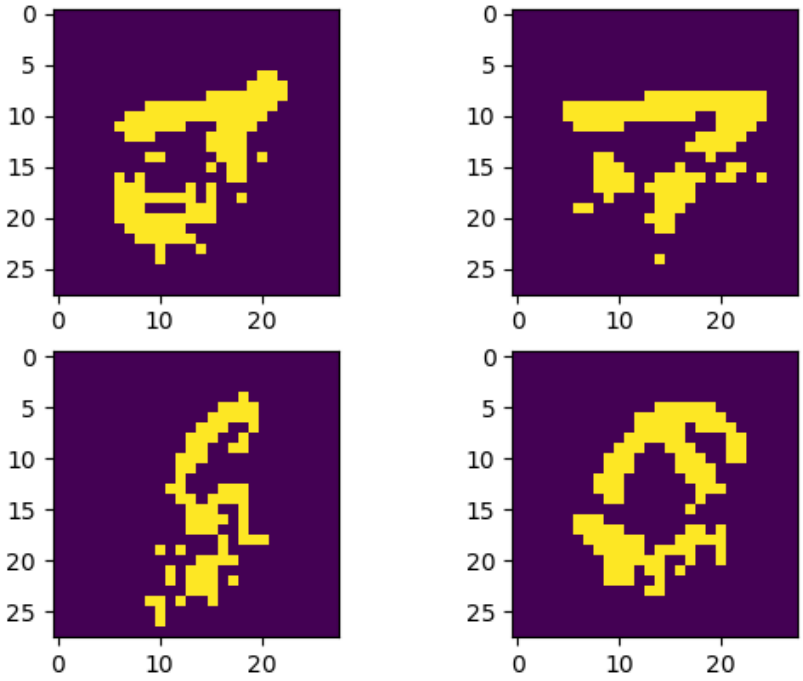


3. Make a variant of the demo where you fit a sigmoid belief network to the data, where the parents of pixel (i, j) are the other pixels in the region $(1 : i, 1 : j)$. Hand in your code and an example of using this model to fill in 4 random test images.

   Answer: The sigmoid DAG could look like this

```
models = Array{Any}(m,m)
include("logReg.jl")
@printf("Fitting...\n")
offset = 2
for i in 15:m
    for j in 1:m
       @show (i,j)

        d = i*j
        Xsub = zeros(n,d)
        k = 1
        for ii in 1:i
            for jj in 1:j
                Xsub[:,k] = X[ii,jj,:]
                k += 1
            end
        end
        models[i,j] = logReg(Xsub[:,1:d-1],Xsub[:,d])
    end
end
```

Some images filled-in with this model look like:



Hint: you may find it helpful to make an m by m array called *models* where element (i, j) contains the model for pixel (i, j). Included in *a4.zip* are *tabular.jl* and *logreg.jl* which implement solving a supervised learning problem using the tabular and sigmoid method (respectively). You may want to debug on a smaller version of the training set, since the runtime of fitting these models is non-trivial (solving 784 supervised learning problems takes time, although you could parallelize to make this go very quickly).

3 Very-Short Answer Questions

Give a short and concise 1-sentence answer to the below questions.

1. What are two reasons that we might want to be able to sample from a distribution?

   Answer: There are many possible answers, like using sampling to form a Monte Carlo estimate, using samples to get an idea of what the model learned, or using samples to try to identify problems with the model.

2. What is the inverse transform method used for?

   Answer: Sampling from 1D distributions where we can compute the inverse CDF.

3. Describe how we could use ancestral sampling to sample from a naive Bayes model.

   Answer: Sample the class from $p(y)$, then independently sample each feature from $p(x_j \mid y)$.

4. What is the cost of generating a sample from a Markov chain of length d with k possible states for each time? What is the cost of decoding?

   Answer: Sampling costs $O(dk)$, decoding costs $O(dk^2)$.

5. Suppose we have a black-box procedure for sampling from a student t distribution. Describe how we could use this black box to get an approximation of the variance of the distribution.

Answer: A basic Monte Carlo approximation would generate a bunch of samples and then compute their variance.

6. What is the difference between inference and decoding in Markov chains?

   Answer: Decoding finds the joint assignment to states of the Markov chain with the highest probability, inference computes the probability of each individual state at each time.

7. What is "explaining away"?

   Answer: When knowing the value of a child in a v-structure (in a DAG) leads to a dependence between the parents.

8. If two variables are not d-separated, are they necessarily depenent? If two variables are d-separated, are they necessarily independent?

   Answer: Not d-separated does not imply dependence, d-separated does imply independence.

9. Why do we enforce that DAG models are acyclic?

   Answer: So that we can use the chain rule to define a valid joint distribution.

10. What is the relationship between multivariate Gaussians and UGMs?

    Answer: Multivariate Gaussians are pairwise UGMs.

11. For each of the following, name a method we covered in class that applies for UGMs:

    (a) Approximate decoding.

    (b) Approximate sampling.

    (c) Approximate inference.

    Answer: Many answers are possible, but you could say ICM, Gibbs sampling (MCMC), and Gibbs sampling (MCMC) or mean-field (variational)

4 Relevant Papers for Project

4.1 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).

4.2 Paper Review

Among your list of 10 papers, choose one paper and write a review of this paper. It makes sense to choose a paper that is closely-related to your project or to choose 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, and whether the claims are appropriately evaluated. 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 a motivation (the authors must have thought it was worth writing for some reason).