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 30 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.

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.

Solution

1 and 2.
Using the script sampleAncestral.m along with the script sampleDiscrete.m to produce k samples in $\mathcal{O}(\log(k))$ we produced a 10000 from the Markov Chain with 30 elements and we also calculated the exact marginals using the script marginalsCK.m. To run the scripts we used marginalsCalculation.m shown below.

```
load viterbiData.mat

```

```
n=10000; %Number of samples

X=sampleAncestral(p0,pT_long,n);
marginals=sum(X==1)/n;
marginals=[marginals;1-marginals];
display('The marginals obtained using Monte Carlo estimation: ')
marginals'

marginals2=marginalCK(p0,pT_long);

display('The marginals obtained using CK equations: ')
marginals2'
```

The output obtained by running the above script is the following

```
>>marginalCalculation

The marginals obtained using Monte Carlo estimation:

ans =

    0.5999    0.4001
    0.4720    0.5280
    0.9035    0.0965
    0.9704    0.0296
    0.7755    0.2245
    0.2263    0.7737
    0.0533    0.9467
    0.3323    0.6677
    0.3163    0.6837
    0.6032    0.3968
    0.3505    0.6495
    0.7586    0.2414
    0.3695    0.6305
    0.1321    0.8679
    0.3646    0.6354
    0.4015    0.5985
    0.7233    0.2767
    0.7879    0.2121
    0.3122    0.6878
    0.6750    0.3250
    0.5848    0.4152
    0.2511    0.7489
    0.4032    0.5968
    0.2293    0.7707
    0.7452    0.2548
    0.6203    0.3797
    0.6039    0.3961
    0.7432    0.2568
    0.4740    0.5260
    0.4347    0.5653
```

```
The marginals obtained using CK equations:

ans =

    0.6000    0.4000
    0.4681    0.5319
    0.8998    0.1002
    0.9718    0.0282
    0.7779    0.2221
    0.2267    0.7733
    0.0542    0.9458
    0.3359    0.6641
    0.3204    0.6796
    0.5902    0.4098
    0.3601    0.6399
    0.7479    0.2521
    0.3726    0.6274
    0.1313    0.8687
    0.3744    0.6256
    0.4103    0.5897
    0.7240    0.2760
    0.7877    0.2123
    0.3147    0.6853
    0.6698    0.3302
    0.5811    0.4189
    0.2541    0.7459
    0.4037    0.5963
    0.2403    0.7597
    0.7378    0.2622
    0.6211    0.3789
    0.6060    0.3940
    0.7419    0.2581
    0.4644    0.5356
    0.4380    0.5620
```

As we can see convergence is slow, since the approximation with 10000 samples gives accurate result to just one significant figure.

3.

Now we proceed to maximize the marginals, i.e. we calculate

$$\max_{x_1,\dots,x_{30}} \mathbb{P}(x_1)\dots\mathbb{P}(x_{30}).$$

To calculate this quantity we used the script marginalDecode.m. The output obtained shows the most likely chain given the maximization problem stated above

```
>>marginalDecode(p0,pT_long)

ans =

  Columns 1 through 13
```

```
     1     2     1     1     1     2     2     2     2     1     2     1     2

  Columns 14 through 26

     2     2     2     1     1     2     1     1     2     2     2     1     1

  Columns 27 through 30

     1     1     2     2
```

The probability to obtain this chain can be easily calculated using

$$\mathbb{P}(1,2,1,\ldots,2,2) = \mathbb{P}(x_1 = 1)\mathbb{P}(x_2 = 2|x_1 = 1)\mathbb{P}(x_3 = 1|x_1 = 1)\cdots\mathbb{P}(x_{30} = 2|x_{29} = 2).$$

The result obtained using the probability transition matrices contained in $pT\_long(:,:,1:29)$ is

$$\mathbb{P}(1,2,1,\ldots,2,2) = 5.0639 \times 10^{-5}.$$

In the next section we will see how this compares with the probability for the chain with highest probability.
4.
Now we calculate the chain with maximum probability using the script viterbiDecode.m. The results are shown below

```
>>viterbiDecode(p0,pT_long)

ans =

  Columns 1 through 13

     1     1     1     1     1     2     2     2     2     1     2     1     2

  Columns 14 through 26

     2     2     2     2     1     2     1     1     2     2     2     1     1

  Columns 27 through 30

     1     1     1     1
```

The probability to get this chain is given by

$$\mathbb{P}(1,1,\ldots,1,1) = 1.7293 \times 10^{-4}.$$

This means that getting the chain with the maximum probability is almost 3.5 times more likely to see it than the chain that is obtained by maximizing the marginal probabilities.

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 $p(x_j|x_d = 1)$.

Solutions

1 and 2
Using the script sampleAncestral.m and using the initial probability $p0$ as $p0 = [1, 0]$ we produce a 10000 samples from the Model where every chain starts at $x_1 = 1$. With these samples we do a Monte Carlo integration to approximate the marginals. Using the same initial probability $p0$ we use the script marginalCK.m to calculate the exact conditional marginals. Below we show the results obtained from the Monte Carlo approximation and the exact calculation of the marginals, where as before the $j - th$ row in the first column is $\mathbb{P}(x_j = 1|x_1 = 2)$ and the second column is $\mathbb{P}(x_j = 2|x_1 = 2)$

```
>>question12
The conditional probability p(xj|x1=2) is given by:

ans =

         0    1.0000
    0.0619    0.9381
    0.9327    0.0673
    0.9747    0.0253
    0.7754    0.2246
    0.2221    0.7779
    0.0600    0.9400
    0.3428    0.6572
    0.3142    0.6858
    0.5965    0.4035
    0.3562    0.6438
    0.7500    0.2500
    0.3777    0.6223
    0.1287    0.8713
    0.3767    0.6233
    0.4081    0.5919
    0.7273    0.2727
    0.7871    0.2129
    0.3094    0.6906
```

```
    0.6746    0.3254
    0.5877    0.4123
    0.2649    0.7351
    0.4004    0.5996
    0.2433    0.7567
    0.7336    0.2664
    0.6164    0.3836
    0.6015    0.3985
    0.7438    0.2562
    0.4595    0.5405
    0.4333    0.5667


The exact univariate conditionals are:

M =

         0    1.0000
    0.0634    0.9366
    0.9297    0.0703
    0.9756    0.0244
    0.7789    0.2211
    0.2265    0.7735
    0.0542    0.9458
    0.3359    0.6641
    0.3204    0.6796
    0.5902    0.4098
    0.3601    0.6399
    0.7479    0.2521
    0.3726    0.6274
    0.1313    0.8687
    0.3744    0.6256
    0.4103    0.5897
    0.7240    0.2760
    0.7877    0.2123
    0.3147    0.6853
    0.6698    0.3302
    0.5811    0.4189
    0.2541    0.7459
    0.4037    0.5963
    0.2403    0.7597
    0.7378    0.2622
    0.6211    0.3789
    0.6060    0.3940
    0.7419    0.2581
    0.4644    0.5356
    0.4380    0.5620
```

As before we can see that the Monte Carlo integration with 10000 sample only gives accuracy of one significant digit.

3.

Now we use the script viterbiDecoding using the same initial probability $p0$ as above, with this we get

```
>>question12
The sequence with highest probability starting with x1=2 is:

ans =

  Columns 1 through 13

     2     2     1     1     1     2     2     2     2     1     2     1     2

  Columns 14 through 26

     2     2     2     2     1     2     1     1     2     2     2     1     1

  Columns 27 through 30

     1     1     1     1
```

4.
To calculate the sequence with highest probability starting with $x_d = 1$ we change the transition probability matrix $p_{ij} = p(x_{30} = j|x_{29} = i)$ from the value

$$pT\_long(:,:,29) = \begin{bmatrix} 0.5972 & 0.4028 \\ 0.2999 & 0.7001 \end{bmatrix}$$

to the value

$$pT\_long(:,:,29) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

With this new value from the transition probability, we can guarantee that regardless the value of x_{29} is, we get $x_{30} = 1$ with probability 1. With this we get that the sequence with highest probability ending in 1 is

```
>>question12
The sequence with highest probability with xd=1 is:

ans =

  Columns 1 through 13

     1     1     1     1     1     2     2     2     2     1     2     1     2

  Columns 14 through 26

     2     2     2     2     1     2     1     1     2     2     2     1     1

  Columns 27 through 30

     1     1     1     1
```

Since the most probable chain end with 1, the chain obtained here is exactly the same as the most likely chain, as expected (compare with part 4, exercise 1.1).
5.
To calculate the conditionals $p(x_j|x_{30} = 1)$ we use the product rule to get

$$\mathbb{P}(x_j|x_{30} = 1) = \frac{\mathbb{P}(x_j, x_{30} = 1)}{\mathbb{P}(x_{30} = 1)}.$$

To calculate this number we generate 10000 samples. To calculate the marginals we used the script sampleReject.m shown below

```
%Script to do the sample reject to calculate conditionals
load viterbiData.mat
n=10000;


Y=sampleAncestral(p0,pT_long,n);
d=size(Y,2);



marginalxd=(sum(Y(:,d)==1)/n);

aux=Y(:,d)==1;
aux=Y(aux,:);

p1=zeros(d,1);
p2=zeros(size(p1));
for (k=1:d)
        temp=aux(:,k)==1;
        p1temp=sum(temp)/n;
        temp=aux(:,k)==2;
        p2temp=sum(temp)/n;

        p1(k)=p1temp/marginalxd;
        p2(k)=p2temp/marginalxd;

end
display('The number of samples accepted: ')
n-size(aux,1)


display('the marginals probabilities are')
[p1 p2]
```

Using this script we get the marginals and the acceptance rate to be

```
>>sampleReject
The number of samples accepted:

ans =

        5618

the marginals probabilities are

ans =

    0.5961    0.4039
    0.4630    0.5370
    0.8927    0.1073
```

```
    0.9717    0.0283
    0.7713    0.2287
    0.2271    0.7729
    0.0568    0.9432
    0.3366    0.6634
    0.3284    0.6716
    0.5869    0.4131
    0.3676    0.6324
    0.7481    0.2519
    0.3681    0.6319
    0.1376    0.8624
    0.3786    0.6214
    0.4103    0.5897
    0.7241    0.2759
    0.7866    0.2134
    0.3183    0.6817
    0.6593    0.3407
    0.5790    0.4210
    0.2556    0.7444
    0.4147    0.5853
    0.2440    0.7560
    0.7378    0.2622
    0.6120    0.3880
    0.5977    0.4023
    0.8021    0.1979
    0.6392    0.3608
    1.0000         0
```

6.
Now we obtain the conditional marginals $p(x_j|x_{30} = 1)$ using backward sampling. In order to do backward sampling we need to know probabilities of the form $\mathbb{P}(x_i|x_{i+1})$. Since we know all probabilities of the form $\mathbb{P}(x_{i+1}|x_i)$ we can calculate

$$\mathbb{P}(x_i|x_{i+1}) = \frac{\mathbb{P}(x_{i+1}|x_i)\mathbb{P}(x_i)}{\mathbb{P}(x_{i+1}}$$

we calculate the marginals using the script marginalCK.m used before in part 2 exercise 1.1. After having found the transition probabilities from future to past we sample chains that $\mathbb{P}(x_{30} = 1) = 1$ we sampled from each term in the product given by

$$\mathbb{P}(x_{30} = 1, x_{29}, \ldots, x_1) = \prod_{j=1}^{29} \mathbb{P}(x_{30-j}|x_{31-j})$$

To implement this samples, we used the script sampleBackwards.m. Then we estimated all the conditional marginals. From a 10000 sample we get the following Monte Carlo estimates

```
X=sampleBackwards(p0,pT_long,10000);
>> [sum(X==1)'/10000 sum(X==2)'/10000]

ans =

    0.5897    0.4103
    0.4606    0.5394
    0.8983    0.1017
```

```
    0.9740    0.0260
    0.7795    0.2205
    0.2205    0.7795
    0.0511    0.9489
    0.3356    0.6644
    0.3181    0.6819
    0.5863    0.4137
    0.3579    0.6421
    0.7531    0.2469
    0.3702    0.6298
    0.1298    0.8702
    0.3685    0.6315
    0.4018    0.5982
    0.7210    0.2790
    0.7947    0.2053
    0.3128    0.6872
    0.6741    0.3259
    0.5823    0.4177
    0.2532    0.7468
    0.4061    0.5939
    0.2376    0.7624
    0.7354    0.2646
    0.6259    0.3741
    0.6155    0.3845
    0.7936    0.2064
    0.6384    0.3616
    1.0000         0
```

7.
Finally we calculate the exact conditional marginals $\mathbb{P}(x_j|x_{30=1})$ by first calculating the functions $M_j(x_j)$ and $V_j(x_j)$ and setting $M_{30}(1) = V_{30} = 1$ along with $M_{30}(2) = V_{30}(2) = 0$. and then we calculate

$$\mathbb{P}(x_j|x_{30} = 1) = \frac{M_j(x_j)V_j(x_j)}{\sum_{x_j} M_j(x_j)V_j(x_j)}.$$

To calculate the marginals we used the script forwardBackwards.m. The results are shown below.

```
>>forwardBackwards(p0,pT_long)

ans =

    0.7332    0.2668
    0.8495    0.1505
    0.9925    0.0075
    0.9823    0.0177
    0.6826    0.3174
    0.0256    0.9744
    0.0541    0.9459
    0.1854    0.8146
    0.4172    0.5828
    0.3238    0.6762
```

0.5978	0.4022
0.6964	0.3036
0.0629	0.9371
0.1944	0.8056
0.7800	0.2200
0.9228	0.0772
0.9607	0.0393
0.6456	0.3544
0.4854	0.5146
0.4718	0.5282
0.3574	0.6426
0.1880	0.8120
0.1859	0.8141
0.2245	0.7755
0.8173	0.1827
0.7944	0.2056
0.8141	0.1859
0.6661	0.3339
0.5625	0.4375
1.0000	0

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.

Solution:

For the ease of notation, in this exercise we are going to use the following notation

1. $x_j := y$
2. $x_{j-1} := x$
3. $w_j := w$
4. $m_j := m$
5. $v_j^2 := \frac{1}{\lambda}$
6. $\mu_{j-1} := \mu$
7. $\sigma_{j-1}^2 = \frac{1}{\tau}$

With this notation we have

$$x \sim \mathcal{N}(\mu, \frac{1}{\tau}),$$
$$y|x \sim \mathcal{N}(wx + m, \frac{1}{\lambda}).$$

To find the prior we use Bayes rule, that can be rewritten as

$$\mathbb{P}(y) = \frac{\mathbb{P}(y|x)\mathbb{P}(x)}{\mathbb{P}(x|y)} \propto \frac{\mathbb{P}(y|x)}{\mathbb{P}(x|y)}. \tag{1}$$

Hence all we need to do is to know how the posterior for x, $\mathbb{P}(x|y)$ is distributed. From Theorem 4.4.1 from Murphy's book we have

$$x|y \sim \mathcal{N}(\mu_{x|y}, \Sigma_{x|y}),$$

where

$$\Sigma_{x|y}^{-1} := \tau + w^2\lambda,$$
$$\mu_{x|y} = \Sigma_{x|y}^{-1}(\lambda w(y - m) + \tau\mu).$$

[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.

By knowing the distribution of $x|y$ and $y|x$ we can use equation (1) to get

$$\mathbb{P}(y) \propto \frac{\exp(-\frac{\lambda}{2}(y-(wx+m))^2}{\exp(-\frac{(x-\mu_{x|y})^2}{2\Sigma_{x|y}})}.$$

This equation can be simplified further if we remove all terms that don't depend on y. By doing this we get

$$\mathbb{P}(y) \propto \frac{\exp(-\frac{\lambda}{2}(y^2-2y(wx+m)))}{\exp(-\frac{1}{2\Sigma_{x|y}}(\mu_{x|y}^2-2\mu_{x|y}x))}.$$

Let us denote $a := \frac{1}{\Sigma_{x|y}}$ and combine the terms in the quotient above to get

$$\mathbb{P}(y) \propto \exp(-\frac{1}{2}(\lambda y^2 - 2\lambda(wx+m)y - a\mu_{x|y}^2 + 2ax\mu_{x|y})).$$

From the definition of $\mu_{x|y}$ is easy to see

$$\begin{aligned} \mu_{x|y} &= \frac{1}{a}(\lambda w(y-m)+\tau\mu), \\ \mu_{x|y}^2 &= a^{-2}(\lambda^2 w^2 (y-m)^2 + 2\lambda w(y-w)\tau\mu + (\tau\mu)^2). \end{aligned}$$

Plugging these values into the posterior for y and keeping only the terms that depend on y we obtain

$$\mathbb{P}(y) \propto \exp(-\frac{1}{2}(\lambda y^2 - 2\lambda(wx+m)y - a(a^{-2}(\lambda^2w^2y^2 - 2\lambda^2w^2my + 2\lambda w\tau\mu y) + 2ax(\frac{1}{a}\lambda wy)))).$$

Rearranging and factoring the above equation reads as

$$\mathbb{P}(y) \propto \exp(-\frac{1}{2}(C_1y^2 + C_2y)),$$

where

$$\begin{aligned} C_1 &:= \lambda - \frac{\lambda^2w^2}{a}, \\ C_2 &:= \frac{2\lambda^2w^2m}{a} - 2\lambda(wx+m) + 2x\lambda w - \frac{2\lambda w\tau\mu}{a}. \end{aligned}$$

If we replace the value of a in C_1 and C_2 and simplify we obtain

$$\begin{aligned} C_1 &= \lambda\tau, \\ C_2 &= -\frac{2\lambda\tau}{\tau+w^2\lambda}(w\mu+m). \end{aligned}$$

Finally if we complete squares in equation (1.3) we get

$$\mathbb{P}(y) \propto \exp(-\frac{C_1}{2}(y+\frac{C_2}{C_1})^2).$$

This means that $y \sim \mathcal{N}(-\frac{C_2}{C_1}, \frac{1}{C_1})$. If we simplify $\frac{C_2}{C_1}$ the result is

$$\frac{C_2}{C_1} = -2(w\mu+m).$$

Hence we finally conclude

$$y \sim \mathcal{N}(2(w\mu+m), \frac{1}{\lambda\tau}).$$

Putting everything in terms of subindices j and $j-1$ we get

$$\begin{aligned} \mu_j &= 2(w\mu+m) = 2(w_j\mu_{j-1}+m_j), \\ \sigma_j^2 &= \frac{1}{\lambda\tau} = (\sigma_{j-1}v_j)^2. \end{aligned}$$

This gives the recursive formula for the mean and variance.

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. $B \perp F$. **True:** We have the v-structure $B \to E \leftarrow C$ with E not seen.
2. $B \perp F \mid A$. **True:** Same reason as before, every possible path from B to F contains $B \to E \leftarrow C$ with E not seen.
3. $B \perp F \mid C$. **True:** In every path from B to F we have the fork $E \leftarrow C \to F$, with C observed.
4. $B \perp F \mid E$. **True:** Boundary conditions.
5. $B \perp F \mid I$. **True:** Same reason as before, every possible path from B to F contains $B \to E \leftarrow C$ with E not seen.
6. $B \perp F \mid J$. **True:** Same reason as before, every possible path from B to F contains $B \to E \leftarrow C$ with E not seen.
7. $B \perp F \mid C, E$. **False:** We are going to prove this by contradiction. That is assume that

$$\mathbb{P}(B, F|C, E) = \mathbb{P}(B|C, E)\mathbb{P}(F|C, E) = \mathbb{P}(B)\mathbb{P}(F|C). \tag{2}$$

Assuming this is true we calculate

$$\mathbb{P}(B, F|C, E) = \frac{\mathbb{P}(B, F, C, E)}{\mathbb{P}(C, E)} = \frac{\mathbb{P}(B)\mathbb{P}(C)\mathbb{P}(F|C)\mathbb{P}(E|B, C)}{\mathbb{P}(C)\mathbb{P}(E|C)}$$

By simplifying and rearranging

$$\mathbb{P}(B, F|C, E) = \underbrace{\mathbb{P}(B)\mathbb{P}(F|C)}_{\mathbb{P}(B,F|C,E) \text{ by eqn. (2)}} \frac{\mathbb{P}(E|B, C)}{\mathbb{P}(E|C)}.$$

This means that

$$\mathbb{P}(E|B, C) = \mathbb{P}(E|C).$$

That is E is independent of B, a contradiction.

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:

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

$$\begin{aligned}
&\text{Dyspnoea}\\
p(D=1|T=1,L=1,B=1) &= 0.90\\
p(D=1|T=1,L=1,B=0) &= 0.70\\
p(D=1|T=1,L=0,B=1) &= 0.85\\
p(D=1|T=1,L=0,B=0) &= 0.65\\
p(D=1|T=0,L=1,B=1) &= 0.82\\
p(D=1|T=0,L=1,B=0) &= 0.60\\
p(D=1|T=0,L=0,B=1) &= 0.80\\
p(D=1|T=0,L=0,B=0) &= 0.10
\end{aligned}$$

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)

Solutions

0.
From the table we have $\mathbb{P}(S = 1) = 0.2$.
1.
By the sum to one restriction we have $\mathbb{P}(S = 0) = 0.8$.
2.
From the table we have $\mathbb{P}(L = 1|S = 1) = 0.1$.
3.
By marginalizing over S we have

$$\begin{aligned}\mathbb{P}(L = 1) &= \sum_S \mathbb{P}(L = 1, S) \\ &= \sum_S \mathbb{P}(L = 1|S)\mathbb{P}(S).\end{aligned}$$

Since we know all the values in the last sum, we conclude

$$\mathbb{P}(L = 1) = 0.01 \times 0.8 + 0.1 \times 0.2 = 0.028.$$

4.
From the table we have

$$\mathbb{P}(X = 1|T = 1, L = 1) = 1.$$

5.
To find the value of $\mathbb{P}(X = 1|T = 1)$ we marginalize over L to get

$$\mathbb{P}(X = 1|T = 1) = \sum_L \mathbb{P}(X = 1|T = 1, L)\mathbb{P}(L|T = 1)$$

All terms in the summand were found before, so we calculate

$$\mathbb{P}(X = 1|T = 1) = 0.98056$$

6.
Now we compute $\mathbb{P}(X=1|T=1,S=1)$ marginalizing over L to get

$$\mathbb{P}(X=1|T=1,S=1)=\sum_L \mathbb{P}(X=1|L,T=1,S=1)\mathbb{P}(L|T=1,S=1)$$

Since L and T are unconditionally independent we get

$$\mathbb{P}(X=1|T=1,S=1)=\sum_L \mathbb{P}(X=1|L,T=1,S=1)\mathbb{P}(L|S=1).$$

We know all the values in the sum, so we get

$$p(X=1|T=1,S=1)=0.982$$

7.
To find $\mathbb{P}(X=1)$, we first need to find the distribution for T. First we calculate

$$\begin{aligned}\mathbb{P}(T=0) &= \sum_V \mathbb{P}(T=0|V)\mathbb{P}(V)\\ &= 0.99\times 0.99+0.95\times 0.01\\ &= 0.9896\end{aligned}$$

This means that $\mathbb{P}(T=1)=0.0104$. With this we can easily calculate

$$\mathbb{P}(X=1)=\sum_L\sum_T \mathbb{P}(X=1|L,T)\mathbb{P}(L)\mathbb{P}(T),$$

where in the last line we used the independence between L and T. Plugging in every value in the sum we conclude

$$\mathbb{P}(X=1)=0.3354.$$

8.
Now we calculate $\mathbb{P}(T=1|X=1)$ via

$$\mathbb{P}(T=1|X=1)=\frac{\mathbb{P}(X=1|T=1)\mathbb{P}(T=1)}{\mathbb{P}(X=1)}$$

Replacing by the numeric value of each probability we get

$$\mathbb{P}(T=1|X=1)=\frac{0.9806\times 0.0104}{0.3354}=0.030406.$$

9.
Since L and T are independent we get

$$\mathbb{P}(T=1|L=1)=\mathbb{P}(T=1)\mathbb{P}(L=1)=2.912e-4.$$

10.
Finally to calculate $\mathbb{P}(T=1|X=1,L=1)$ we use Bayes rule, unconditional independence between T and L and product rule for $\mathbb{P}(X,L)$ to get

$$\mathbb{P}(T=1|X=1,L=1)=\frac{\mathbb{P}(X=1|T=1,L=1)\mathbb{P}(T=1)}{\mathbb{P}(L=1)\mathbb{P}(X=1|L=1)}=\frac{1\times 0.0104}{0.028\times 0.90104}=0.412222,$$

where we use marginalization over T to calculate the term $\mathbb{P}(X=1|L=1)$.

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 and 3, 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 parts 2 and 3 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 3, 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).

Solutions

1.
First we fitted an inhomogeneous Markov chain, where given a pixel x_j we calculated the transition probability from x_{j-1} as

$$\mathbb{P}(x_j|x_{j-1}) = \frac{\text{Number of times we have a transition from } x_{j-1} \text{ to } x_j}{\text{Number of times we started at } x_j}.$$

We trained our model using the script demoInhomogeneousMC.m. We used the model on 4 random images from the test set. The results are shown below

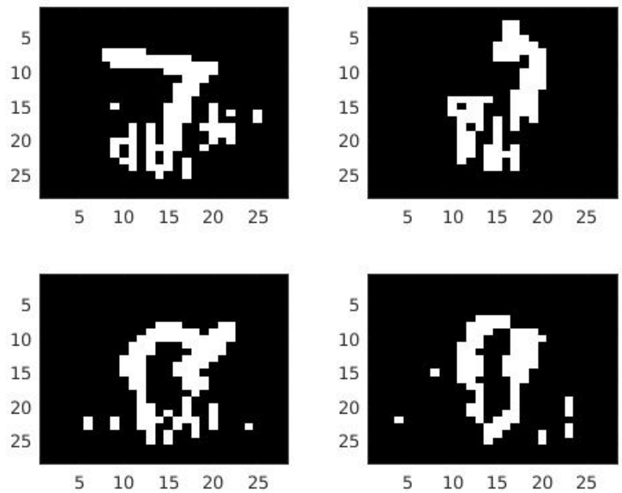

Figure 1: Four samples from an inhomogeneous Markov Chain model

2 and 3.
In this part of the exercise we trained two DAGs where each pixel had 8 parents and 15 parents. The parents for the pixel (i, j) where the pixels in the region $((i-2) : i, (j-2) : j)$ and $((i-3) : i, (j-3) : j)$ respectively. The scripts used to create the model are demoDAG8.m and demoDAG15.m. The results from this different models are shown below.

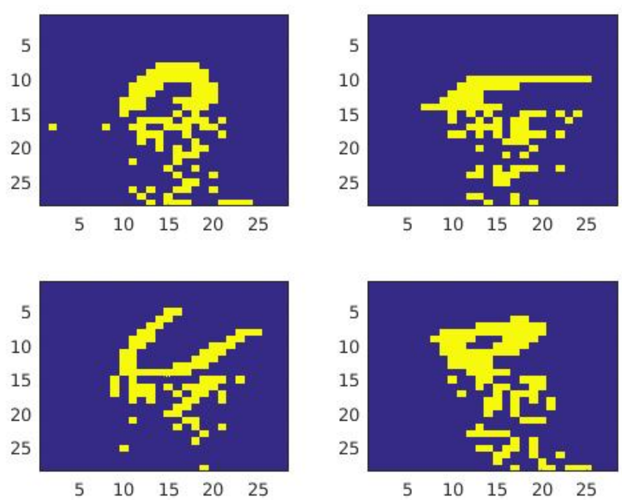

Figure 2: Four sample images from a DAG model with 8 parents.

The results for the DAG with 15 parents are

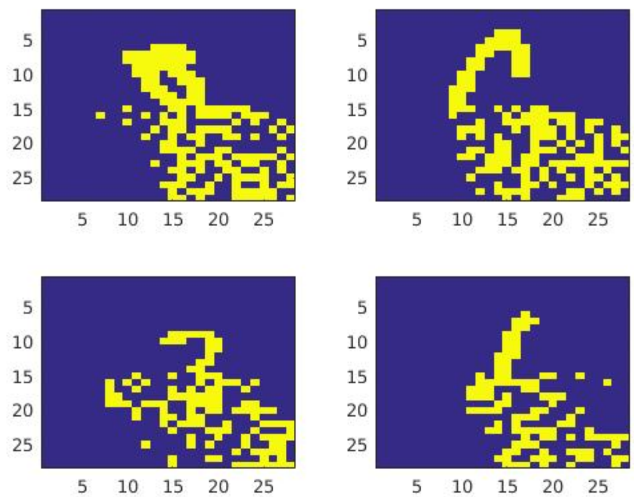

Figure 3: Four sample images from a DAG model with 15 parents.

It is interesting to see that often the pixel located at $(28, 28)$ has value 1 despite all training images have the value of 0 at that location. To understand why this is case consider the parent located at $(25, 25)$. There are 16 training images with 1 in that entry, this means $\mathbb{P}(x_{(25,25)} = 1) \neq 0$. Also pixels $(25, 26)$ has five images with that characteristic and $(26, 25)$ five images. Hence there are different combinations of vectors $x_{pa((28,28))} \in \{0, 1\}^{15}$ such that

$$\mathbb{P}(x_{(28,28)=1} | x_{pa((28,28))}) \neq 0.$$

More precisely the key is that the parent $x_{(25,25)}$ has non-zero marginal probability, hence if $x_{(25,25)} = 1$ for example

$$\mathbb{P}(x_{(28,28)} = 1, x_{(28,27)}, \ldots, x_{(25,25)} = 1) = \underbrace{\mathbb{P}(x_{(25,25)} = 1)}_{\neq 0} \underbrace{\mathbb{P}(x_{(26,26)} = 1 | x_{(25,25)} = 1) \cdots \mathbb{P}(x_{(28,28)} | x_{pa((28,28))})}_{\neq 0}.$$

This represents the fact that since the (grand) parent (25,25) has non-zero entries there are sequences ending with $x_{(28,28)} = 1$ with non-zero probability.

4.

Finally we trained a sigmoid belief network where the parents of the pixel located at the point (i, j) are the pixels in the region $(1 : i, 1 : j)$. we used the script demoSigmoid.m (it takes a couple of minutes to load). The results are shown below.

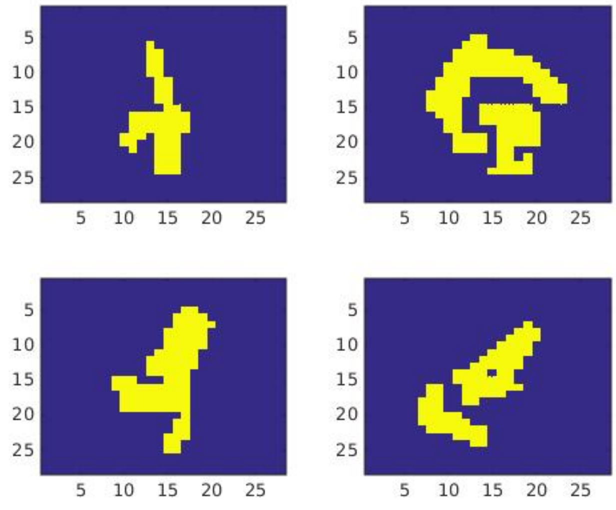

Figure 4: Four samples from a sigmoid belief network.

3. Paper Review

Finding Relevant Papers

The list of the top ten most important papers found so far are

[1] Kim, Kyoung-jae. "Financial time series forecasting using support vector machines." Neurocomputing 55.1 (2003): 307-319.

[2] Tsai, C. F., and S. P. Wang. "Stock price forecasting by hybrid machine learning techniques." Proceedings of the International MultiConference of Engineers and Computer Scientists. Vol. 1. No. 755. 2009.

[3] Yeh, Chi-Yuan, Chi-Wei Huang, and Shie-Jue Lee. "A multiple-kernel support vector regression approach for stock market price forecasting." Expert Systems with Applications 38.3 (2011): 2177-2186.

[4] Yoo, Paul D., Maria H. Kim, and Tony Jan. "Machine learning techniques and use of event information for stock market prediction: A survey and evaluation." Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on. Vol. 2. IEEE, 2005.

[5] Hassan, Md Rafiul, and Baikunth Nath. "Stock market forecasting using hidden Markov model: a new approach." Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on. IEEE, 2005.

[6] Oh, Kyong Joo, Tae Yoon Kim, and Sungky Min. "Using genetic algorithm to support portfolio optimization for index fund management." Expert Systems with Applications 28.2 (2005): 371-379.

[7]Freitas, Fabio D., Alberto F. De Souza, and Ailson R. de Almeida. "Prediction-based portfolio optimization model using neural networks." Neurocomputing 72.10 (2009): 2155-2170.

[8] Milosevic, Nikola. "Equity forecast: Predicting long term stock price movement using machine learning." arXiv preprint arXiv:1603.00751 (2016).

[9] Gabaix, Xavier, et al. "A theory of power-law distributions in financial market fluctuations" Nature 423.6937 (2003): 267-270.

[10] Oliveira, Nuno, Paulo Cortez, and Nelson Areal. "The impact of microblogging data for stock market prediction: Using Twitter to predict returns, volatility, trading volume and survey sentiment indices." Expert Systems with Applications 73 (2017): 125-144.

Paper Review

The paper chosen to do the review is paper number [5] if the above list.

Summary

The paper Stock market forecasting using hidden Markov model: a new approach, by Hassan et.al. Even though is a little bit updated, is the first paper that provides an implements the use of Hidden Markov Models (HMM) to the task of stock market forecasting. HMMs were successfully applied before to in areas like speech recognition DNA sequencing etc. But before this paper was published HMM had not been applied to stock market prediction. In particular the authors of the mentioned paper chose to apply the method in the market of airlines.

The authors propose to apply HMM with four features. The features they selected where

- Opening price.
- Closing price.
- Highest price.
- Lowest price.

The HMM takes in this four inputs and gives as an output the next day's closing price. To evaluate the performance their method, the authors gathered the information from 4 different airlines. They used as a training set the data of these airlines from the dates of December 18 2002 to 29 September 2004 to predict the closing stock price on the 30 of September. In the paper the authors compare their results with the mainstream method at that time: neural networks (NN). Based on their result, they were able to outperform the results obtained when using (NN) to predict the closing price. With this overview of the paper, we now proceed to evaluate the strengths and weaknesses of the paper.

Strengths

1. The paper proposes for the first time a novel method to tackle the long time and unsolved problem of predicting the prices in the stock market.
2. The explanation are straightforward and is easy to understand the main idea of the method they use.
3. The results obtained look promising and outperform a well known method by that time. They also mention where this research could lead to by talking about their future work.

Weaknesses

1. The authors chose just four features under the argument that it is difficult to choose features in the setting of HMM for financial prediction and they don't mention any intention to improve on that.
2. They don't explain why they chose the airlines stock market. Is it because they try other different markets and they didn't get as nice results as with airlines? Is because they had more data from this market? Was completely at random the choice?
3. In the same spirit they don't explain why they chose to measure the performance of their model in predicting the closing price of the stocks. Was because when they try to predict other features they failed?
4. Finally, this might be just nitpicking, but the graphs they show were made in Excel and I find it unprofessional to do that in the academic environment, since there is a wide variety of (free) softwares capable to plot in a much more aesthetic way.