Name:	Student ID:
Pillar: ISTD/ESD/EPD/ASD	

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning
Final Exam
2016 Term 6

Date: 16 Dec 2016
Start Time: 3:00 P.M.
Duration: 2 hours

Instructions:

1. Write your name, student ID, and pillar information at the top of this page.
2. This paper consists of 4 main questions and 12 printed pages.
3. The problems are not necessarily in order of difficulty. We recommend that you scan through all the questions first, and then decide on the order to answer them.
4. Write your answers in the space provided.
5. You may refer to your one-sided A4-sized cheat sheet.
6. You are allowed to use non-programmable calculators.
7. You may NOT refer to any other material.
8. You may NOT access the Internet.
9. You may NOT communicate via any means with anyone (aside from the invigilators).

For staff's use:

Qs 1	/8
Qs 2	/25
Qs 3	/20
Qs 4	/12
Total	**/65**

Cheatsheet.

2 ↗

4 ↗

Question 1. (8 points)
Please indicate whether the following statements are true (**T**) or false (**F**).

1. In reinforcement learning, the Q-learning algorithm will not update the robot/agent's belief about the Q values until the robot/agent takes an action. *(1 point)*

   Answer : T

2. The goal of learning a Markov decision process (MDP) is to learn a policy that specifies for each state a specific action to take. *(1 point)*

   Answer : T

3. The Bayesian networks are generative probabilistic models. Such generative models can be represented graphically using ~~un~~directed graphs consisting of nodes and undirected arcs. *(1 point)*

   Answer : F

4. In Markov decision process discussed in class, we can use value iteration algorithm to find an optimal policy. Specifically, we use the algorithm to find the optimal value for each state s, and then based on the values we compute the optimal Q-values for each state-action pair (s, a). Next we obtain the optimal action for each state. *(1 point)*

   Answer : T

5. The hard EM for the hidden Markov model is analogous to the k-means algorithm for clustering. Essentially, at the ~~M~~-step (E-step), the algorithm assigns to each input observation sequence $\mathbf{x}$ a specific $\mathbf{y}$ sequence. *(1 point)*

   Answer : F

6. The forward-backward algorithm and the Viterbi algorithm share the same time complexity. *(1 point)* $n(0) = n(nT^2)$

   Answer : T

7. In Bayesian network structure learning, if we use the log-likelihood as the objective when selecting the optimal Bayesian network structures, we will almost always end up with some very complex Bayesian networks with many edges. This is mainly because a more complex model tends to explain the data better. *(1 point)*

   Answer : T

8. To optimize the model parameters for the naive Bayes or the hidden Markov model in a supervised setting where we have tuples $(\mathbf{x}, \mathbf{y})$ in the training set, we can typically derive closed-form solutions to the model parameters, and procedures such as stochastic gradient descent will not be required. *(1 point)*

   Answer : T

Question 2. (25 points)
In this problem, we would like to look at the hidden Markov model (HMM).

(a) Assume that we have the following data available for us to estimate the model parameters:

6A
5B
4C

	State sequence		Observation sequence
START	(A, B, C, A, B)	STOP	(y, z, x, y, z)
START	(B, C, A, C)	STOP	(y, z, x, y)
START	(C, A, B)	STOP	(x, y, x)
START	(A, A, B)	STOP	(z, y, x)

HMM

Clearly state what are the parameters associated with the HMM. Under the maximum likelihood estimation (MLE), what would be the optimal model parameters? Fill up the following transition and emission probability tables. Use the space below to clearly show how **each emission parameter** is estimated exactly. *(10 points)*

$a_{u,v}$ $u\backslash v$	**A**	**B**	**C**	STOP
START	2/4 = 0.5	1/4 = 0.25	1/4 = 0.25	0/4 = 0
A	1/6	4/6	1/6	0/6 = 0
B	0/5 = 0	0/5 = 0	2/5	3/5
C	3/4 = 0.75	0/4 = 0	0/4 = 0	1/4 = 0.25

$b_u(o)$ $u\backslash o$	**x**	**y**	**z**
A	1/6	4/6 = 2/3	1/6
B	2/5 = 0.4	1/5 = 0.2	2/5 = 0.4
C	2/4 = 0.5	1/4 = 0.25	1/4 = 0.25

Emission Probabilities are estimated as $b_u(o) = \frac{\text{count}(u \to o)}{\text{count}(u)}$

$$b_A(x) = \frac{\text{count}(A \to x)}{\text{count}(A)} = 1/6$$

$$b_A(y) = \frac{\text{count}(A \to y)}{\text{count}(A)} = 4/6 = 2/3$$

$$b_A(z) = \frac{\text{count}(A \to z)}{\text{count}(A)} = 1/6$$

$$b_B(x) = \frac{\text{count}(B \to x)}{\text{count}(B)} = 2/5$$

$$b_B(y) = \frac{\text{count}(B \to y)}{\text{count}(B)} = 1/5$$

$$b_B(z) = \frac{\text{count}(B \to z)}{\text{count}(B)} = 2/5$$

$$b_C(x) = \frac{\text{count}(C \to x)}{\text{count}(C)} = 2/4 = 0.5$$

$$b_C(y) = \frac{\text{count}(C \to y)}{\text{count}(C)} = 1/4 = 0.25$$

$$b_C(z) = \frac{\text{count}(C \to z)}{\text{count}(C)} = 1/4 = 0.25$$

(b) Now, consider in the test phase, you are given the following new observation sequence and the following parameters, find the most probable state sequence using the Viterbi algorithm discussed in class. Clearly present the steps that lead to your final answer. *(5 points)*

	A	**B**	**C**	STOP
START	0.2	0.3	0.5	0.0
A	0.2	0.2	0.4	0.2
B	0.2	0.1	0.2	0.5
C	0.4	0.3	0.1	0.2

	x	**y**	**z**
A	0.4	0.5	0.1
B	0.4	0.1	0.5
C	0.2	0.6	0.2

State sequence	Observation sequence
(?, ?)	(**y**, **z**)

BASE CASE : $\pi(0,START)=1$

RECURSIVE CASE

$\pi(1,A)=\pi(0,START)\cdot a_{START,A}\cdot b_A(y) = 1\cdot 0.2\cdot 0.5 = 0.1$

$\pi(1,B)=\pi(0,START)\cdot a_{START,B}\cdot b_B(y) = 1\cdot 0.3\cdot 0.1 = 0.03$

$\pi(1,C)=\pi(0,START)\cdot a_{START,C}\cdot b_C(y) = 1\cdot 0.5\cdot 0.6 = 0.3$

$\pi(2,A)=\max_{u\in T}\{\pi(1,u)\cdot a_{u,A}\cdot b_A(z)\}$

$= \max\{(0.1)(0.2)(0.1),\ (0.03)(0.2)(0.1),\ (0.3)(0.4)(0.1)\} = 0.012$

$\pi(2,B)=\max_{u\in T}\{\pi(1,u)\cdot a_{u,B}\cdot b_B(z)\}$

$= \max\{(0.1)(0.2)(0.5),\ (0.03)(0.1)(0.5),\ (0.3)(0.3)\overset{C}{(0.5)}\} = 0.045$

$\pi(2,C)=\max_{u\in T}\{\pi(1,u)\cdot a_{u,C}\cdot b_C(z)\}$

$= \max\{(0.1)(0.4)(0.2),\ (0.03)(0.2)(0.2),\ (0.3)(0.1)(0.2)\} = 0.008$

$\pi(3,STOP)=\max_{v\in T}\{\pi(2,v)\cdot a_{v,STOP}\}$

$= \max\{(0.012)(0.2),\ (0.045)\overset{B}{(0.5)},\ (0.008)(0.2)\} = 0.0225$

Backtracking: $y_2^* = B$, $y_1^* = C$ → optimal sequence: C, B

RECURSIVE CASE **BASE CASE** : $\pi(0,START)=1$

$\pi(1,A)=\pi(0,START)\cdot a_{START,A}\cdot b_A(y) = 1\cdot 0.2\cdot 0.5 = 0.1$

$\pi(2,A)=\max_{u\in T}\{\pi(1,u)\cdot a_{u,A}\cdot b_A(z)\}$

$= \max\{(0.1)(0.2)(0.1),\ (0.03)(0.2)(0.1),\ (0.3)(0.4)(0.1)\} = 0.012$

$\pi(3,STOP)=\max_{v\in T}\{\pi(2,v)\cdot a_{v,STOP}\}$

$= \max\{(0.012)(0.2),\ (0.045)\overset{B}{(0.5)},\ (0.008)(0.2)\} = 0.0225$

Backtracking: $y_2^* = B$, $y_1^* = C$ → optimal sequence: C, B

(c) In the Viterbi algorithm we are interested in finding the optimal state sequence, or most probable state sequence using the following formula:

$$(s_1^*, s_2^*) = \arg\max_{s_1,s_2} P(s_1, s_2|o_1 = \mathbf{y}, o_2 = \mathbf{z}) \tag{1}$$

Sometimes we are also interested in finding the "worst" state sequence – the least probable state sequence. In other words, we are interested in:

$$(s_1^*, s_2^*) = \arg\min_{s_1,s_2} P(s_1, s_2|o_1 = \mathbf{y}, o_2 = \mathbf{z}) \tag{2}$$

This procedure is useful in some advanced structured prediction problems. Briefly and clearly explain how to perform such decoding efficiently by modifying the Viterbi algorithm. *(5 points)*

using Dynamic Programming

BASE CASE : $\pi(0,u) = \begin{cases} 1 \ ; & \text{if } u = \text{START} \\ 0 \ ; & \text{otherwise} \end{cases}$

RECURSIVE CASE : $\pi(k,v) = \min_{u \in T} \{ \pi(k-1,u) \cdot a_{u,v} \cdot b_v(k) \}$

TRANSITION : $\min_{v \in T} \{ \pi(n,v) \cdot a_{v,STOP} \}$

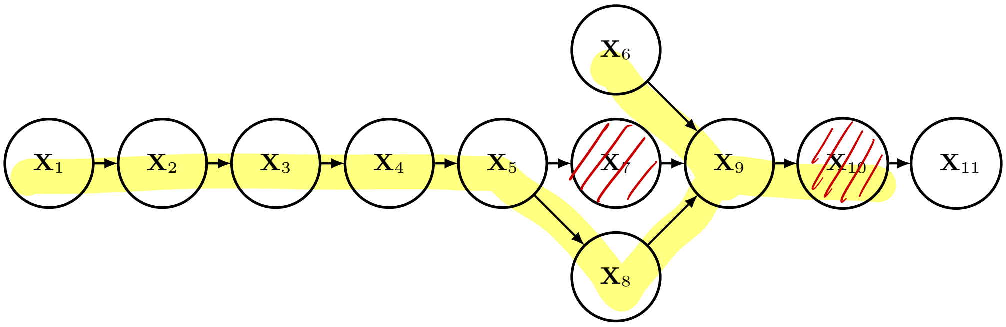
$(s_1^*, s_2^*) \arg\min_{s_1,s_2} a_{START,s_1} \cdot b_{s_1}(y) \cdot a_{s_1,s_2} \cdot b_{s_2}(z)) \cdot a_{s_2,STOP}$

Find the least probable state sequence for the observation sequence $(\mathbf{y}, \mathbf{z})$. *(5 points)*

Using the workings from part (b).

Least probable is (B, A)

Question 3. (20 points)
Consider the following Bayesian network, where we have 11 variables. Assume all variables are binary, taking values from $\{1, 2\}$.



(a) Answer if X_1 and X_6 are independent or dependent of each other if no other variable is given. *(3 points)*

Answer: Independent. (*independent* or *dependent*)

Reason: There is no open path from X_1 to X_6. Hence, by use of the Bayes' Ball Algorithm, X_1 and X_6 are independent.

(b) Answer if X_1 and X_6 are independent or dependent of each other if both X_7 and X_{10} are given. *(3 points)*

Answer: Dependent. (*independent* or *dependent*)

Reason: By use of Bayes' Ball Algorithm, given X_{10} is known, there is an open path from X_1 to X_6 that allows both to be independent. The open path is

$X_1 - X_2 - X_3 - X_4 - X_5 - X_8 - X_9 - X_{10} - X_9 - X_6$

X_3 independent of X_2

	$\mathbf{X}_1$ = 1	$\mathbf{X}_1$ = 2
	0.5	0.5

$\mathbf{X}_1$	$\mathbf{X}_2$ = 1	$\mathbf{X}_2$ = 2
1	0.2	0.8
2	0.3	0.7

$\mathbf{X}_2$	$\mathbf{X}_3$ = 1	$\mathbf{X}_3$ = 2
1	0.3	0.7
2	0.3	0.7

$\mathbf{X}_3$	$\mathbf{X}_4$ = 1	$\mathbf{X}_4$ = 2
1	0.1	0.9
2	0.5	0.5

$\mathbf{X}_4$	$\mathbf{X}_5$ = 1	$\mathbf{X}_5$ = 2
1	0.5	0.5
2	0.6	0.4

	$\mathbf{X}_6$ = 1	$\mathbf{X}_6$ = 2
	0.6	0.4

$\mathbf{X}_5$	$\mathbf{X}_7$ = 1	$\mathbf{X}_7$ = 2
1	0.2	0.8
2	0.3	0.7

$\mathbf{X}_5$	$\mathbf{X}_8$ = 1	$\mathbf{X}_8$ = 2
1	0.8	0.2
2	0.7	0.3

$\mathbf{X}_6$	$\mathbf{X}_7$	$\mathbf{X}_8$	$\mathbf{X}_9$ = 1	$\mathbf{X}_9$ = 2
1	1	1	0.8	0.2
1	1	2	0.1	0.9
1	2	1	0.9	0.1
1	2	2	0.7	0.3
2	1	1	0.3	0.7
2	1	2	0.2	0.8
2	2	1	0.2	0.8
2	2	2	0.9	0.1

$\mathbf{X}_9$	$\mathbf{X}_{10}$ = 1	$\mathbf{X}_{10}$ = 2
1	0.8	0.2
2	0.8	0.2

$\mathbf{X}_{10}$	$\mathbf{X}_{11}$ = 1	$\mathbf{X}_{11}$ = 2
1	0.7	0.3
2	0.8	0.2

X_{10} independent of X_9

(c) Now, assume the probability table for each node is shown above. How many independent parameters are involved in this Bayesian network? Clearly show the number of independent parameters for each individual node. *(4 points)*

No of free parameters $= 1(X_1) + 2\times1(X_2) + 2\times1(X_3) + 2\times1(X_4) + 2\times1(X_5) + 1(X_6)$
$+ 2\times1(X_7) + 2\times1(X_8) + 2\times2\times2\times1(X_9) + 2\times1(X_{10}) + 2\times1(X_{11})$

$= 1 + 2 + 2 + 2 + 2 + 1 \rightarrow 10$
$+ 2 + 2 + 8 + 2 + 2 \rightarrow 16$

$= 26$

(d) Calculate the following conditional probability *(5 points)*

$$P(\mathbf{X}_3 = 2|\mathbf{X}_4 = 1)$$

(*Hint: find a short answer.*)

$P(X_3=2|X_4=1)$
$= \dfrac{P(X_3=2, X_4=1)}{P(X_4=1)}$
$= \dfrac{P(X_4=1, X_3=2)}{P(X_4=1)}$
$= \dfrac{P(X_4=1|X_3=2)P(X_3=2)}{P(X_3=1)P(X_4=1|X_3=1)+P(X_3=2)P(X_4=1|X_3=2)}$
$= \dfrac{(0.7)(0.5)}{(0.3)(0.1)+(0.7)(0.5)}$
$= 0.92$

(e) Calculate the following conditional probability *(5 points)*

$$P(\mathbf{X}_5 = 2|\mathbf{X}_3 = 1, \mathbf{X}_{11} = 2, \mathbf{X}_1 = 1)$$

(*Hint: find a short answer. The values in some of the probability tables may reveal some useful information.*)

(b) $P(X_5 = 2 | X_3 = 1, X_{11} = 2, X_1 = 1)$

$= \dfrac{P(X_5 = 2 | X_3 = 1)}{P(\quad X_3 = 1)}$

$= \dfrac{P(X_3 = 1) P(X_5 = 2 | X_4) P(X_4 | X_3 = 1)}{0.3}$

$= \dfrac{P(X_3=1) P(X_5=2|X_4=1) P(X_4=1|X_3=1) + P(X_3=1) P(X_5=2|X_4=2) P(X_4=2|X_3=1)}{0.3}$

$= \dfrac{(0.3)(0.1)(0.5) + (0.3)(0.9)(0.4)}{0.3} = \dfrac{0.123}{0.3} = 0.41$

X_{10} independent of X_9

$\rightarrow$ X_{11} independent of X_5

X_3 independent of X_2

$\rightarrow$ X_2 independent of X_1

$\rightarrow$ X_5 independent of X_1

Question 4. (12 points)

Consider the following Markov decision process (MDP). It has states $\{0, 1, 2\}$. In every state, you can take one of two possible actions: A or B. State 0 is the terminal state (*i.e.*, once the agent reaches that state, it stays there no matter what action it takes).

The transition probabilities for action A and B are given as:

$0.5 \times 0^2 + 0.5 \times 1^2 + 0.0 \times 2^2 = 0.5$

$1.0 \times 0^2 + 0.0 \times 1^2 + 0.0 \times 2^2 = 0$

$R(s, a, s') = s'^2$.

$T(s, A, s')$ $s \backslash s'$	0	1	2
0	1.0	0.0	0.0
1	0.5	0.5	0.0
2	0.5	0.0	0.5

$T(s, B, s')$ $s \backslash s'$	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0

For example, $T(2, A, 0) = 0.5$, $T(2, B, 1) = 1.0$.

The reward function is defined as $R(s, a, s') = s'^2$. The discount factor is $\gamma = 0.6$.

Let us consider the Q-value iteration algorithm.

1. Suppose we initialize $Q_0^*(s, a) = 0$ for all $s \in \{0, 1, 2\}$ and $a \in \{A, B\}$. Evaluate the Q-values $Q_1^*(s, a)$ after exactly one iteration of the Q-Value Iteration Algorithm. Write your answers in the table below.

$1.0 \times 0^2 + 0.0 \times 1^2 + 0.0 \times 2^2 = 0$

$0.5 \times 0^2 + 0.5 \times 1^2 + 0.0 \times 2^2 = 0.5$

$0.5 \times 0^2 + 0.0 \times 1^2 + 0.5 \times 2^2 = 2$

$1.0 \times 0^2 + 0.0 \times 1^2 + 0.0 \times 2^2 = 0$

$1.0 \times 0^2 + 0.0 \times 1^2 + 0.0 \times 2^2 = 0$

$0.0 \times 0^2 + 1.0 \times 1^2 + 0.0 \times 2^2 = 1$

	$s = 0$	$s = 1$	$s = 2$
A	0	0.5 ✓	2 ✓
B	0	0	1

2. What is the policy that we would derive from $Q_1^*(s, a)$? Answer by filling in the action that should be taken at each state in the table below.

$s = 1$	$s = 2$
A	A

3. What are the values $V_1^*(s)$ corresponding to $Q_1^*(s, a)$?

$s = 0$	$s = 1$	$s = 2$
0	0.5	2

4. Will the policy change after the second iteration? If your answer is "yes", briefly describe how.

No.

Name: Vivek kalyan	**Student ID:** 1001457
Pillar: ISTD/ESD/EPD/ASD	

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning
Final Exam
2016 Term 6

Date: 16 Dec 2016
Start Time: 3:00 P.M.
Duration: 2 hours

Instructions:

1. Write your name, student ID, and pillar information at the top of this page.
2. This paper consists of 4 main questions and 12 printed pages.
3. The problems are not necessarily in order of difficulty. We recommend that you scan through all the questions first, and then decide on the order to answer them.
4. Write your answers in the space provided.
5. You may refer to your one-sided A4-sized cheat sheet.
6. You are allowed to use non-programmable calculators.
7. You may NOT refer to any other material.
8. You may NOT access the Internet.
9. You may NOT communicate via any means with anyone (aside from the invigilators).

For staff's use:

Qs 1	/8
Qs 2	/25
Qs 3	/20
Qs 4	/12
Total	**/65**

HMM

Sequence labeling problem - map sequence of observations to sequence of tags
- $x_1, x_2 \dots x_n \to y_1, y_2 \dots y_n$
- needs to take into account previous tags

Joint probability distribution

$p(x_1 \dots x_n, y_0, y_1, \dots y_{n+1}) = \prod_{i=1}^{n+1} p(y_i|y_{i-1})\, p(y_i|x_i)$ y_0 = START, y_{n+1} = STOP

$\mathcal{T}$ is set of states including START and STOP states

O is set of observation symbol

$\theta = \langle a, b \rangle$

$a_{u,v} \equiv p(y_{next}=v|y_{curr}=u) = \frac{count(u,v)}{count(u)}$

$b_u(o) \equiv p(x=o|y=u) = \frac{count(u\to o)}{count(u)}$

Therefore, probability for each possible (x,y) pair is given by:

$p_\theta(x,y) = p(x,y;\theta)$

$= a_{START,y_1} \cdot a_{y_n,STOP} \cdot \prod_{i=1}^{n-1} a_{y_i,y_{i+1}} \cdot \prod_{i=1}^{n} b_{y_i}(x_i)$

Viterbi Algorithm - dynamic programming algorithm

Input: $x = x_1 \dots x_n$ and θ

Output: $\arg\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_0, y_1, \dots, y_{n+1};\theta)$

Joint probability for the first k tags:

$r(y_1, \dots, y_k) = \prod_{i=1}^{k} a_{y_{i-1},y_i} \prod_{i=1}^{k} b_{y_i}(x_i)$

$p(x_1 \dots x_n, y_0, y_1, \dots y_{n+1}) = r(y_1 \dots y_n) \cdot a_{y_n,y_{n+1}} = r(y_1 \dots y_n) \cdot a_{y_n,STOP}$

Let $S(k,v)$ be the set of tag sequences $y_1 \dots y_k$ such that $y_k = v$

$\Rightarrow S(k,v)$ is set of all sequences of length k whose last tag is v

The dynamic programming algorithm will calculate the following recursively (forward)

$\pi(k,v) = \max_{(y_1 \dots y_k) \in S(k,v)} r(y_1 \dots y_k)$

⇒ solving the maximization problem partially over the tags $y_1 \dots y_{k-1}$ with constraint that we use tag v for y_k

Base Case:

$\pi(0,u) = \begin{cases} 1 & \text{if } u = START \\ 0 & \text{otherwise} \end{cases}$

Moving forward recursively: for any $k \in \{1, \dots, n\}$

$\pi(k,v) = \max_{u \in \mathcal{T}} \{\pi(k-1,u) \cdot a_{u,v} \cdot b_v(x_k)\}$

Transition from y_n to STOP:

$\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_0, y_1 \dots y_n, y_{n+1}) = \max_{v \in \mathcal{T}} \{\pi(n,v) \cdot a_{v,STOP}\}$

⇒ time complexity → $O(n|\mathcal{T}|^2)$, n = length, $\mathcal{T}$ = # of tags

$y_n^* = \arg\max_v \{\pi(n,v) \cdot a_{v,STOP}\}$

$y_{n-1}^* = \arg\max_u \{\pi(n-1,u) \cdot a_{u,y_n^*}\}$ and so on

Unsupervised Learning → given unlabelled observation sequences xs and a collection of tags but not the tag sequences ys

eg. $\mathcal{T} = \{A, B\}$, $O = \{a, b, c\}$

output sequences: [(a,b), (a,c), (a,b)]

HARD EM (similar to k-means for clustering)

E-step:
- each data point x is strictly assigned to a single y
- → each observation sequence strictly assigned to a single state sequence
- given model params and observation sequences: find corresponding state sequences.
- 1. Run Viterbi algorithm to find most probable state sequence

M-step:
- given observation sequences and state sequences, find model parameters
- 1. Parameter estimation using MLE

$a_{u,v} = \frac{count(u,v)}{count(u)}$ $b_u(o) = \frac{count(u\to o)}{count(u)}$

SOFT EM:

E-step:
- evaluate posterior probability over possible tag sequences
- $Count(x^i, y, u\to v)$: # of times a transition from state u to v occurs in tag sequence y given obs $x^{(i)}$
- 1. Reassign partial memberships for each $x^{(i)}$

$Count(u,v) = \sum_i \sum_y p(y|x^{(i)})\, Count(x^i, y, u \to v)$

M-step:
- 1. Re-estimate transition probabilities based on expected counts

$a_{u,v} = \frac{count(u,v)}{\sum_{v'} count(u,v')}$ where $\sum_{v'} a_{uv'} = 1$

Problem → calculating sum over possible tag sequences in E-step has exponential hidden state sequences y.

Suppose we can efficiently calculate marginal probabilities:

$Count(u;v) = \sum_i \sum_y p(y|x^{(i)})\, Count(x^i, y, u\to v) = \sum_{i=1}^{m} \sum_{j=0}^{n} p(y_j=u, y_{j+1}=v|x^{(i)};\theta)$

Inference

How to efficiently compute a simpler marginal posterior probability

$p(y_j=u|x;\theta) = \sum_{y: y_j=u} p(y|x;\theta) = \frac{p(x, y_j=u;\theta)}{p(x;\theta)} = \frac{p(x_1, x_2, \dots x_{j-1}, y_j=u, x_j, \dots x_n;\theta)}{p(x_1, x_2, \dots x_n;\theta)}$

Numerator can be written as:

$p(x_1, x_2 \dots x_{j-1}, y_j=u, x_j, \dots x_n;\theta) = p(x_1, x_2 \dots x_{j-1}, y_j=u) \cdot p(x_j, x_{j+1}, \dots, x_n|x_1, x_2 \dots x_{j-1}, y_j=u;\theta)$

in HMM, given y_j, $x_j \dots x_n$ are conditionally independent of $x_1 \dots x_{j-1}$ ∴

$p(x_1, x_2 \dots x_{j-1}, y_j=u, x_j, \dots x_n;\theta) = p(x_1, x_2 \dots x_{j-1}, y_j=u) \cdot p(x_j, x_{j+1}, \dots, x_n|y_j=u;\theta)$

Forward Probability ($\alpha_u(j)$)

$\alpha_u(j) = p(x_1, x_2 \dots x_{j-1}, y_j=u;\theta)$

⇒ probability of emitting the symbols $x_1 \dots x_{j-1}$ and ending in state u with position j without yet emitting the corresponding output symbol

Backward Probability ($\beta_u(j)$)

$\beta_u(j) = p(x_j, x_{j+1} \dots x_n|y_j=u;\theta)$

⇒ probability of emitting symbols $x_j \dots x_n$ and final STOP state given we began in state u in position j.

$p(x;\theta) = p(x_1, x_2, \dots x_n;\theta) = \sum_u p(x_1, x_2, \dots, x_{j-1}, y_j=u, x_j, \dots, x_n;\theta)$

$= \sum_u p(x_1, x_2, \dots, x_{j-1}, y_j=u;\theta) \cdot p(x_j, x_{j+1}, \dots, x_n|y_j=u;\theta) = \sum_u \alpha_u(j)\beta_u(j)$

These give us:

$p(y_j=u, y_{j+1}=v|x;\theta) = \frac{p(x_1, x_2 \dots x_{j-1}, y_j=u;\theta) \cdot p(y_{j+1}=v|y_j=u) \cdot p(x_j|y_j=u) \cdot p(x_{j+1} \dots x_n|y_{j+1}=v;\theta)}{p(x;\theta)}$

$= \frac{\alpha_u(j) \cdot a_{u,v} \cdot b_u(x_j) \cdot \beta_v(j+1)}{\sum_{u'} \alpha_{u'}(i)\beta_{u'}(i)}$

Sum over all possible tag sequences that include the transition $y_j=u$ and $y_{j+1}=v$ and generates the observations, divided by sum of all sequences that generate the observations

→ obtain relative probability of transition, relative to all the alternatives given the observations (posterior)

Forward Backward Algorithm

For a state sequence $y_1, y_2 \dots y_n$ there is:
- a path through the graph that has sequence of states START, $\langle 1, y_1\rangle, \dots, \langle n, y_n\rangle$, STOP
- path associated with state sequence $y = y_1 \dots y_n$ has score: $p(x,y;\theta)$
- $\alpha_u(j)$ is sum of scores of all paths from START to state $\langle j, u\rangle$
- $\beta_u(j)$ is sum of scores of all paths from state $\langle j, u\rangle$ to STOP
- Given input sequence $x_1, \dots, x_n$ for $u \in 1, \dots, N$, $j \in 1, \dots, n$, the forward and backward probability can be calculated recursively.

Forward probability

$\alpha_u(j) = p(x_1, \dots x_{j-1}, y_j = u)$

Base case:

$a_u(1) = a_{START,u}\ \forall u \in$

Recursive case:

$a_u(j+1) = \sum_v \alpha_v(j)\, a_{v,u}\, b_v(x_j)\ \forall u \in 1, \dots, N-1, j = 1, \dots, n-1$

Backward probability

$\beta_u(j) = p(x_j, \dots, x_n|y_j = u;\theta)$

Base case:

$\beta_u(n) = a_{u,STOP}\, b_u(x_n)\ \forall u \in 1, \dots, N-1$

Recursive case:

$\beta_u(j) = \sum_v a_{u,v}\, b_u(x_j)\, \beta_v(j+1)\ \forall u \in 1, \dots, N-1, j = n-1, \dots, 1$

Transition Parameters

$Count(u;v) = \sum_{i=1}^{m} Count^{(i)}(u;v)$, $Count(u) = \sum_{i=1}^{m} Count^{(i)}(u)$

$Count^{(i)}(u;v) = \sum_{j=0}^{n} P(y_j = u, y_{j+1} = v|x^{(i)};\theta^{t-1})$

$= \sum_{j=0}^{n} \frac{1}{Z}\alpha_u(j)\, b_u(x_j^{(i)})\, a_{u,v}\, \beta_v(j+1)$, $Z = \sum_{u'} a_{u'}(j)\beta_{u'}(j)$

$Count^{(i)}(u) = \sum_{j=1}^{n} p(y_j = u|x^{(i)};\theta^{t-1})$

$= \sum_{j=1}^{n} \frac{1}{Z}\alpha_u(j)\beta_u(j)$, $Z = \sum_{u'} \alpha_{u'}(j)\beta_{u'}(j)$

Emisson Parameters

$Count(u \to o) = \sum_{i=1}^{m} Count^{(i)}(u \to o)$

$Count^{(i)}(u \to o) = \sum_{j=1}^{n} p(y_j = u, x_j^{(i)} = o|x^{(i)};\theta^{t-1})$

$= \sum_{j\ s.t.\ x_j^{(i)} = o} \frac{1}{Z}\alpha_u(j)\beta_u(j)$, $Z = \sum_{u'} \alpha_{u'}(j)\beta_{u'}(j)$

⇒ can compute E-step of soft EM

Max-marginal Decoding

$p(y_i = u|x_1, \dots, x_n;\theta) = \frac{p(x_1, \dots, x_{i-1}, y_u, x_i, \dots, x_n)}{p(x_1, \dots x_n;\theta)}$

$= \frac{\alpha_u(i)\beta_u(i)}{\sum_v \alpha_v(j)\beta_v(j)}$

$y_i^* = \arg\max_u p(y_i = u|x_1 \dots x_n;\theta) = \arg\max_u \frac{\alpha_u(i)\beta_u(i)}{\sum_v a_v(j)\beta_v(j)}$

$= \arg\max_u \alpha_u(i)\beta_u(i)$

→ Viterbi is often better

BN

Joint distribution → Product of conditional probabilities given its parents

→ if variable has no parents (eg. X_1), then probability is just $P(X_1 = x_1)$

Marginal Independence

$P(X_1 = x_1, X_2 = x_2) = \sum_{x_3} P(X_1 = x_1)P(X_2 = x_2)P(X_3 = x_3|X_1 = x_1, X_2 = x_2)$

$= P(X_1 = x_1)P(X_2 = x_2)$

X_1 and X_2 are marginally independent of each other: if we don't know the value of X_3, there is nothing that ties the coin flips together

Induced dependence

However, if we know the value of X_3, X_1 and X_2 become dependent ↗ possible for them to still be independent? (based on probability)

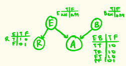
Explaining away

E: Earthquake

B: Burglary

A: Alarm

R: Report



$P(E=e, B=b, A=a, R=r) = P(E=e)P(B=b)P(A=a|E=e, B=b)P(R=r|E=e)$ [product of variable given its parents]

If we observe alarm went off - at least one of E=T or B=T should have occurred (most likely not both)

To calculate $P(B=T|A=T)$, first find $P(B=b, A=T)$

$P(B=b, A=T) = \sum_{e\in\{T,F\}} \sum_{r\in\{T,F\}} P(E=e)P(B=b)P(R=r|E=e)P(A=a|E=e, B=b)$

$= \sum_{e\in\{T,F\}} P(E=e)P(B=b)P(A=a|E=e, B=b) \underbrace{\sum_{r\in\{T,F\}} P(R=r|E=e)}_{=1}$

$= P(B=b)\sum_{e\in\{T,F\}} P(E=e)P(A=a|E=e, B=b)$

$P(B=T|A=T) = \frac{P(B=T, A=T)}{\sum_{b\in T,F} P(B=b, A=T)}$

$= \frac{0.01 \cdot [0.01 \cdot 1 + 0.99 \cdot 1]}{0.01 + [0.01 \cdot 0 + 0.99 \cdot 1]}$ = slightly above 0.5 ⇒ because of chance of both B=T and E=T.

If we now hear a report about earthquake → we now know that E=T, removing any evidence about B=T

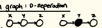
∴ $P(B=T|A=T, R=T) = P(B=T) = 0.01$

$P(E=T|A=T, R=T) = 1$

⇒ R and B are dependent given A=T (induced dependence)

Indepence from graph: D-seperation

Chain:



⇒ shaded means value of variable is known

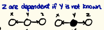
X and Z dependent? ✓ ✗

$P(Z|X,Y) = \frac{P(X,Y,Z)}{P(X,Y)} = \frac{P(X)P(Y|X)P(Z|Y)}{P(X)P(Y|X)} = P(Z|Y)$

⇒ X and Z are independent given Y

⇒ X and Z are dependent if Y is not known

Common Cause:



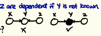
X and Z dependent? ✓ ✗

$P(Z|X,Y) = \frac{P(X,Y,Z)}{P(X,Y)} = \frac{P(Y)P(X|Y)P(Z|Y)}{P(Y)P(X|Y)} = P(Z|Y)$

⇒ X and Z are independent given Y

⇒ X and Z are dependent if Y is not known

Explaining away:



X and Z dependent? ✗ ✓

$P(X,Z) = \sum_Y P(X)P(Z)P(Y|X,Z) = P(X)P(Z)\sum_Y P(Y|X,Z) = P(X)P(Z)$

⇒ X and Z are dependent given Y

⇒ X and Z are independent without knowing Y

Joint Distribution:

$P(X_1 = x_1, \dots X_d = x_d) = \prod_{i=1}^{d} P(X_i = x_i|X_{pa_i} = x_{pa_i}) = \prod_{i=1}^{d} \theta_i(x_i|x_{pa_i})$

⇒ $\theta_i(x_i|x_{pa_i})$ is the probability table for variable i

Number of free parameters:

$\dim(G) = \sum_{i=1}^{d} (r_i - 1)\prod_{j\in pa_i} r_j$

Log-likelihood of data D for a graph G (i.e. how well does graph G explain data D)

$\ell(D;\theta;G) = \sum_{t=1}^{m} \log\left[\prod_{i=1}^{d} \theta_i(x_i^{(t)}|x_{pa_i}^{(t)})\right] = \sum_{t=1}^{m}\sum_{i=1}^{d} \log\theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) = \sum_{i=1}^{d}\left[\sum_{t=1}^{m} \log\theta_i(x_i^{(t)}|x_{pa_i}^{(t)})\right]$

Using values from the table $\theta_i(x_i|x_{pa_i})$

$\sum_{t=1}^{m} \log\theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) = \sum_{x_i, x_{pa_i}} \underbrace{Count(x_i, x_{pa_i} \text{ in } D)}_{\text{(\# of observation in Data D for which } X_i = x_i \text{ and } X_{pa_i} = x_{pa_i})} \cdot \log\theta_i(x_i|x_{pa_i})$

Therefore the corresponding maximum likelihood parameter estimate for $\theta_i(x_i|x_{pa_i})$:

$\hat{\theta}_i(x_i|x_{pa_i}) = \frac{Count((x_i, x_{pa_i}) \text{ in } D)}{Count((x_{pa_i}) \text{ in } D)}$, $x_i \in \{1, \dots r_i\}$

$Count((x_{pa_i}) \text{ in } D) \to \sum_{x_i'} Count((x_i', x_{pa_i}) \text{ in } D)$

Learning the graph structure

We cannot use only the maximum likelihood to determine the graph structure

∵ it favours graphs that have more parameters to fit the data

Therefore, we use the Bayesian Information Criterion (BIC)

$BIC(D;\hat{\theta}, G) = \ell(D;\hat{\theta}, G) - \frac{\dim(G)}{2}\log(m)$

$\dim(G) = \sum_{i=1}^{d} (r_i - 1)\prod_{j\in pa_i} r_j$ → # of free parameters

We can also write it as:

$Score(i|pa_i;D) = \sum_{t=1}^{m} \log\hat{\theta}_i(x_i^{(t)}|x_{pa_i}^{(t)}) - \frac{1}{2}\left[(r_i - 1)\prod_{j\in pa_i} r_j\right]\log(m)$

$BIC(D;\hat{\theta};G) = \sum_{i=1}^{d} score(i|pa_i;D)$

⇒ to maximise BIC score: local search / dynamic programming

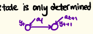
complete data → cond. prob estimates for each variable, parent selection scores for each variable → decomposable scoring func. for graphs → highest scoring acyclic graph

RL

HMM - use states y_t to encode position of the robot and observations x_t to capture sensor readings

process is markov as transition to the next state is only determined by the current state.

$T(i,j) = p(y_{t+1} = j|y_t = i)$



$T(i,k,j) = P(y_{t+1} = j|y_t = i, a_t = k)$, $i \to j$ with action k

Rewards: associated with states: $R(s_t)$

→ can also take into account action that brings robot to next state: $R(s_{t-1}, a_{t-1}, s_t)$

Utility function: long term reward for a robot when it takes actions

To prevent infinite rewards (loops):

1. Assume robot has finite horizon (after N steps utility value does not change)

$U([s_0, s_1 \dots s_N \dots s_{N+k}]) = U([s_0, s_1 \dots s_N])\ \forall k \geq 1$

2. Discounted Rewards (Has finite value even for infinite sequences) ✓

$U([s_0, s_1 \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots = \sum_{t=0}^{\infty} \gamma^t R(s_t)$

$U([s_0, s_1 \dots]) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$

$\Big\}\ 0 \leq \gamma < 1$

→ larger γ means longer sequences

Policy π: function that maps each state to an action

goal → find optimal policy that maximizes utility → action chosen to maximize discounted rewards

-0.01	-0.01	-0.01	+1
-0.01		-0.01	-1
-0.01	-0.01	-0.01	-0.01

→	→	→	+1
↑		←	-1
↑	←	←	↓

$R(s) = -0.01$

-2	-2	-2	+1
-2		-2	-1
-2	-2	-2	-2

→	→	→	+1
↑		→	-1
↑	→	→	↑

$R(s) = -2$

Markov Decision Process → provided with:
- set of states S
- set of actions A
- transition probability function $T(s,a,s') = p(s'|s,a)$
- Reward function $R(s,a,s')$ or $R(s)$

Reinforcement Learning → provided with:
- set of states S
- set of actions A

Value Iteration - find an optimal policy using MDP.

$\pi(s)$ - particular policy that specifies action we should take at state s.

$V^\pi(s)$ - value of state s under policy π

$Q^\pi(s,a)$ - value of state s and action a under policy π

$\pi^*(s)$ - action to take in state s that gives highest utility

$V^*(s)$ - value of state s under optimal policy $\pi^*(s)$

$Q^*(s,a)$ - value of state s and action a under optimal policy π^*

value: → value of rewards for acting optimally from that state

These values are related by:

$V^*(s) = \max_a Q^*(s,a) = Q^*(s, \pi^*(s))$

$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

$= \sum_{s'} T(s, \pi^*(s), s')[R(s, \pi^*(s), s') + \underbrace{\gamma V^*(s')}_{\text{recursive}}]$

Value Iteration Algorithm
- Start $V_0^*(s) = 0$, for all $s \in S$
- Given V_i^*, calculate values for all states $s \in S$ (depth i+1)

$V_{i+1}^* \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i^*(s')]$

- Repeat until convergence (V_{i+1}^* is almost V_i^* for all states)
- Optimal Policy

$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

$\pi^*(s) = \arg\max_a Q^*(s,a)$

Reinforcement Learning (II)

$V^*(s) = \max_a Q^*(s,a) = Q^*(s, \pi^*(s))$ - ①

$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')$ - ② ①→②

$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_a Q^*(s,a)$

Q-value iteration algorithm
- Start with $Q_0^*(s,a) = 0$ for all $s \in S$, $a \in A$
- Given $Q_i^*(s,a)$, calculate the Q-values for all states (depth i+1) and for all actions a

$Q_{i+1}^*(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_i^*(s',a')]$

- Repeat until convergence
- Optimal policy

$\pi^*(s) = \arg\max_a Q^*(s,a)$

Reinforcement Learning - rewards and transitions are not known prior

Two options: 1. Try to learn model, transition probabilities, rewards and solve for optimal policy using MDP algorithms

2. Learn Q-values directly

Model based Learning:

- assume we can collect information about transitions involving any state s and action a.

→ Learn T and R through experience

$T(s,a,s') = \frac{count(s,a,s')}{count(s,a)}$

$R(s,a,s') = \frac{\sum_t R_t(s,a,s')}{count(s,a,s')}$ → reward observed for the t^{th} time when $s \to s'$ with a } might differ due to noisy rewards ∵ avg

Model free Learning:

$\pi^*(s) = \arg\max_a Q^*(s,a)$ → Q-values can be learned recursively (assume we know T & R)

Compute the expected value of function $f(x)$: $E[f(x)] = \sum_x p(x)f(x)$
- Model based computation - estimate $p(x)$ from samples and then compute expectation

$x_i \approx p(x)$, $i = 1, \dots, k$

$\hat{p}(x) = \frac{count(x)}{k}$

$E[f(x)] \approx \sum_x \hat{p}(x)f(x)$

- model-free estimation - estimate expectation directly from samples

$x_i \approx p(x)$, $i = 1, \dots, k$

$E[f(x)] \approx \frac{1}{k}\sum_{i=1}^{k} f(x_i)$

→ apply model free learning approach to the estimation of Q-values

$Q_{i+1}^*(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_i^*(s',a')]$

→ repeatedly take one action at a time, observe reward for next state

Sample 1: $R(s,a,s_1') + \gamma \max_{a'} Q_i(s_1', a')$

⋮

Sample k: $R(s,a,s_k') + \gamma \max_{a'} Q_i(s_k', a')$

Average the samples to obtain the Q-value estimate.

$Q_{i+1}(s,a) \leftarrow \frac{1}{k}\sum_{k=1}^{k} [R(s,a,s_k') + \gamma \max_{a'} Q_i(s_k', a')]$

In practice, we observe states only when we move

Update function: (observed k-1 samples so far, now updating for k^{th} sample

$Q_{new}(s,a) \leftarrow \frac{(k-1)Q_{old}(s,a) + R(s,a,s_k') + \gamma \max_{a'} Q_{old}(s_k', a')}{k}$

$Q(s,a) \leftarrow Q(s,a) + \frac{1}{k}[R(s,a,s_k') + \gamma \max_{a'} Q(s_k', a') - Q(s,a)]$

Q Learning Algorithm
- Collect a sample: s, a, s' and R(s,a,s')
- Update Q values - incoporate new sample to running average over samples

$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$

$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

} learning rate $\alpha = \frac{1}{k}$

Update has same convergence conditions as the gradient ascent algorithm.

(s,a): each sample, can assign seperate learning rate for each case, $\alpha = \alpha_k(s,a)$, k is # of (s,a) so far.

$\sum_k \alpha_k(s,a) \to \infty$ and $\sum_k \alpha_k^2(s,a) < \infty$

Question 1. (8 points)
Please indicate whether the following statements are true (**T**) or false (**F**).

1. In reinforcement learning, the Q-learning algorithm will not update the robot/agent's belief about the Q values until the robot/agent takes an action. *(1 point)*

   Answer : T ✓

2. The goal of learning a Markov decision process (MDP) is to learn a policy that specifies for each state a specific action to take. *(1 point)*

   Answer : T ✓

3. The Bayesian networks are generative probabilistic models. Such generative models can be represented graphically using undirected graphs consisting of nodes and undirected arcs. *(1 point)*

   Answer : F ✓

4. In Markov decision process discussed in class, we can use value iteration algorithm to find an optimal policy. Specifically, we use the algorithm to find the optimal value for each state s, and then based on the values we compute the optimal Q-values for each state-action pair (s, a). Next we obtain the optimal action for each state. *(1 point)*

   Answer : T ✓

5. The hard EM for the hidden Markov model is analogous to the k-means algorithm for clustering. Essentially, at the M-step, the algorithm assigns to each input observation sequence $\mathbf{x}$ a specific $\mathbf{y}$ sequence. *(1 point)*

   Answer : T ✗

6. The forward-backward algorithm and the Viterbi algorithm share the same time complexity. *(1 point)*

   Answer : T ✓

7. In Bayesian network structure learning, if we use the log-likelihood as the objective when selecting the optimal Bayesian network structures, we will almost always end up with some very complex Bayesian networks with many edges. This is mainly because a more complex model tends to explain the data better. *(1 point)*

   Answer : T ✓

8. To optimize the model parameters for the naive Bayes or the hidden Markov model in a supervised setting where we have tuples $(\mathbf{x}, \mathbf{y})$ in the training set, we can typically derive closed-form solutions to the model parameters, and procedures such as stochastic gradient descent will not be required. *(1 point)*

   Answer : T ✓

Question 2. (25 points)
In this problem, we would like to look at the hidden Markov model (HMM).

(a) Assume that we have the following data available for us to estimate the model parameters:

State sequence	Observation sequence
$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{A}, \mathbf{B})$	$(\mathbf{y}, \mathbf{z}, \mathbf{x}, \mathbf{y}, \mathbf{z})$
$(\mathbf{B}, \mathbf{C}, \mathbf{A}, \mathbf{C})$	$(\mathbf{y}, \mathbf{z}, \mathbf{x}, \mathbf{y})$
$(\mathbf{C}, \mathbf{A}, \mathbf{B})$	$(\mathbf{x}, \mathbf{y}, \mathbf{x})$
$(\mathbf{A}, \mathbf{A}, \mathbf{B})$	$(\mathbf{z}, \mathbf{y}, \mathbf{x})$

Clearly state what are the parameters associated with the HMM. Under the maximum likelihood estimation (MLE), what would be the optimal model parameters? Fill up the following transition and emission probability tables. Use the space below to clearly show how each **emission parameter** is estimated exactly. *(10 points)*

$a_{u,v}$ $u\backslash v$	A	B	C	STOP
START	2/4	1/4	1/4	0/4
A	1/6	4/6	1/6	0/6
B	0	0	2/5	3/5
C	3/4	0	0	1/4

$b_u(o)$ $u\backslash o$	x	y	z
A	1/6	4/6	1/6
B	2/5	1/5	2/5
C	2/4	1/4	1/4

Emisson : $p(x=o|y=u) = \frac{\text{count}(u\to o)}{\text{count}(u)}$

$b_A(x) = \frac{\text{count}(A\to x)}{\text{count}(A)} = \frac{1}{6}$

$b_A(y) = \frac{\text{count}(A\to y)}{\text{count}(A)} = \frac{4}{6}$

$b_A(z) = \frac{\text{count}(A\to z)}{\text{count}(A)} = \frac{1}{6}$

$b_B(x) = \frac{\text{count}(B\to x)}{\text{count}(B)} = \frac{2}{5}$

$b_B(y) = \frac{\text{count}(B\to y)}{\text{count}(B)} = \frac{1}{5}$

$b_B(z) = \frac{\text{count}(B\to z)}{\text{count}(B)} = \frac{2}{5}$

$b_C(x) = \frac{\text{count}(C\to x)}{\text{count}(C)} = \frac{2}{4}$

$b_C(y) = \frac{\text{count}(C\to y)}{\text{count}(C)} = \frac{1}{4}$

$b_C(z) = \frac{\text{count}(C\to z)}{\text{count}(C)} = \frac{1}{4}$

(b) Now, consider in the test phase, you are given the following new observation sequence and the following parameters, find the most probable state sequence using the Viterbi algorithm discussed in class. Clearly present the steps that lead to your final answer. *(5 points)*

	A	**B**	**C**	STOP
START	0.2	0.3	0.5	0.0
A	0.2	0.2	0.4	0.2
B	0.2	0.1	0.2	0.5
C	0.4	0.3	0.1	0.2

	x	**y**	**z**
A	0.4	0.5	0.1
B	0.4	0.1	0.5
C	0.2	0.6	0.2

State sequence	Observation sequence
(?, ?)	(**y**, **z**)

$$\arg\max_{u,v} \; a_{START,u} \cdot b_u(y) \cdot a_{u,v} \cdot b_v(z) \cdot a_{v,STOP}$$

A, A : $0.2 \cdot 0.5 \cdot 0.2 \cdot 0.1 \cdot 0.2 = 4\times10^{-4}$

A, B : $0.2 \cdot 0.5 \cdot 0.2 \cdot 0.5 \cdot 0.5 = 5\times10^{-3}$

A, C : $0.2 \cdot 0.5 \cdot 0.4 \cdot 0.2 \cdot 0.2 = 1.6\times10^{-3}$

B, A : $0.3 \cdot 0.1 \cdot 0.2 \cdot 0.1 \cdot 0.2 = 1.2\times10^{-4}$

B, B : $0.3 \cdot 0.1 \cdot 0.1 \cdot 0.5 \cdot 0.5 = 7.5\times10^{-4}$

B, C : $0.3 \cdot 0.1 \cdot 0.2 \cdot 0.2 \cdot 0.2 = 2.4\times10^{-4}$

C, A : $0.5 \cdot 0.6 \cdot 0.4 \cdot 0.1 \cdot 0.2 = 2.4\times10^{-3}$

C, B : $0.5 \cdot 0.6 \cdot 0.3 \cdot 0.5 \cdot 0.5 = 2.25\times10^{-2}$

C, C : $0.5 \cdot 0.6 \cdot 0.1 \cdot 0.2 \cdot 0.2 = 1.2\times10^{-3}$

Most probable state sequence : C, B

(c) In the Viterbi algorithm we are interested in finding the optimal state sequence, or most probable state sequence using the following formula:

$$(s_1^*, s_2^*) = \arg\max_{s_1,s_2} P(s_1, s_2 | o_1 = \mathbf{y}, o_2 = \mathbf{z}) \tag{1}$$

Sometimes we are also interested in finding the "worst" state sequence – the least probable state sequence. In other words, we are interested in:

$$(s_1^*, s_2^*) = \arg\min_{s_1,s_2} P(s_1, s_2 | o_1 = \mathbf{y}, o_2 = \mathbf{z}) \tag{2}$$

This procedure is useful in some advanced structured prediction problems. Briefly and clearly explain how to perform such decoding efficiently by modifying the Viterbi algorithm. *(5 points)*

Using dynamic programming:

Base $\pi(0,u) = \begin{cases} 1 & \text{if } u = \text{START} \\ 0 & \text{otherwise} \end{cases}$

Recursive

$$\pi(K,v) = \min_{u \in T} \{ \pi(K-1,u) \cdot a_{u,v} \cdot b_v(K) \}$$

transition

$$\min_{v \in T} \{ \pi(n,v) \cdot a_{v,\text{STOP}} \}$$

For s_1^* and s_2^*

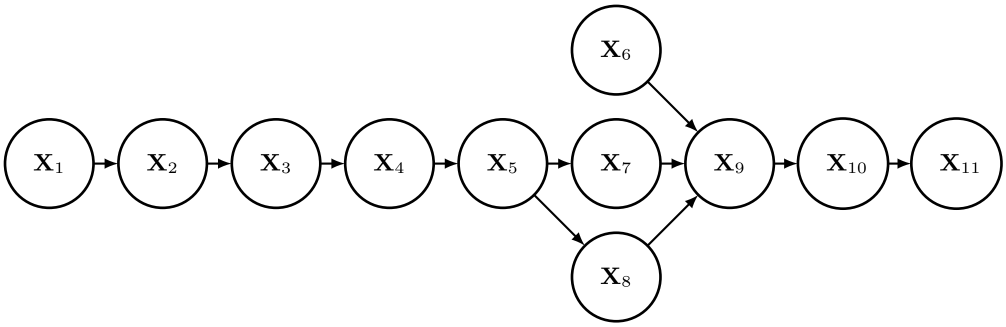
$(s_1^*, s_2^*) \arg\min_{s_1, s_2} \quad a_{\text{START},s_1} \cdot b_{s_1}(y) \cdot a_{s_1,s_2} \cdot b_{s_2}(z) \cdot a_{s_2,\text{STOP}}$

Find the least probable state sequence for the observation sequence $(\mathbf{y}, \mathbf{z})$. *(5 points)*

Using the workings from part (b) -

Least probable is (B, A)

Question 3. (20 points)
Consider the following Bayesian network, where we have 11 variables. Assume all variables are binary, taking values from $\{1, 2\}$.



(a) Answer if X_1 and X_6 are independent or dependent of each other if no other variable is given. *(3 points)*

Answer: Independent (*independent* or *dependent*)

Reason:
Applying the bayes ball algorithm, we cannot roll the ball from X_1 to X_6.

→ path is closed

(b) Answer if X_1 and X_6 are independent or dependent of each other if both X_7 and X_{10} are given. *(3 points)*

Answer: Independent (*independent* or *dependent*)

Reason:
Applying the bayes ball algorithm, still there is no way to roll the ball from X_1 to X_6.

→ path is closed X Wrong

	$\mathbf{X}_1$ = 1	$\mathbf{X}_1$ = 2
	0.5	0.5

$\mathbf{X}_1$	$\mathbf{X}_2$ = 1	$\mathbf{X}_2$ = 2
1	0.2	0.8
2	0.3	0.7

$\mathbf{X}_2$	$\mathbf{X}_3$ = 1	$\mathbf{X}_3$ = 2
1	0.3	0.7
2	0.3	0.7

$\mathbf{X}_3$	$\mathbf{X}_4$ = 1	$\mathbf{X}_4$ = 2
1	0.1	0.9
2	0.5	0.5

$\mathbf{X}_4$	$\mathbf{X}_5$ = 1	$\mathbf{X}_5$ = 2
1	0.5	0.5
2	0.6	0.4

	$\mathbf{X}_6$ = 1	$\mathbf{X}_6$ = 2
	0.6	0.4

$\mathbf{X}_6$	$\mathbf{X}_7$	$\mathbf{X}_8$	$\mathbf{X}_9$ = 1	$\mathbf{X}_9$ = 2
1	1	1	0.8	0.2
1	1	2	0.1	0.9
1	2	1	0.9	0.1
1	2	2	0.7	0.3
2	1	1	0.3	0.7
2	1	2	0.2	0.8
2	2	1	0.2	0.8
2	2	2	0.9	0.1

$\mathbf{X}_5$	$\mathbf{X}_7$ = 1	$\mathbf{X}_7$ = 2
1	0.2	0.8
2	0.3	0.7

$\mathbf{X}_5$	$\mathbf{X}_8$ = 1	$\mathbf{X}_8$ = 2
1	0.8	0.2
2	0.7	0.3

$\mathbf{X}_9$	$\mathbf{X}_{10}$ = 1	$\mathbf{X}_{10}$ = 2
1	0.8	0.2
2	0.8	0.2

$\mathbf{X}_{10}$	$\mathbf{X}_{11}$ = 1	$\mathbf{X}_{11}$ = 2
1	0.7	0.3
2	0.8	0.2

(c) Now, assume the probability table for each node is shown above. How many independent parameters are involved in this Bayesian network? Clearly show the number of independent parameters for each individual node. *(4 points)*

$X_1 = 2-1 = 1$

$x_2 = (2-1)\cdot 2 = 2$

$x_3 = x_4 = x_5 = x_7 = x_8 = x_{10} = x_{11} = (2-1)\cdot 2 = 2.$

$x_6 = 2-1 = 1$

$x_9 = (2-1)\cdot 8 = 8.$

$2\cdot 1 + 8\cdot 2 + 8$

$= \underline{\underline{26}}$

(d) Calculate the following conditional probability *(5 points)*

$$P(\mathbf{X}_3 = 2|\mathbf{X}_4 = 1)$$

(*Hint: find a short answer.*)

$P(x_4=1) = 0.3\cdot 0.1 + 0.7\cdot 0.5 = \underline{0.38}$

$P(x_3=2|x_4=1) = \dfrac{P(x_3=2, x_4=1)}{P(x_4=1)} = \dfrac{0.7\cdot 0.5}{0.38}$

$= \underline{0.921}$

(e) Calculate the following conditional probability *(5 points)*

$$P(\mathbf{X}_5 = 2|\mathbf{X}_3 = 1, \mathbf{X}_{11} = 2, \mathbf{X}_1 = 1)$$

(*Hint: find a short answer. The values in some of the probability tables may reveal some useful information.*)

$P(X_5 = 2 | X_3 = 1, X_{11} = 2, X_1 = 1)$

$= P(X_5 = 2 | X_3 = 1)$

$P(X_5 = 2, X_3 = 1) = 0.1 \cdot 0.5 + 0.9 \cdot 0.4$

$= 0.41$

Question 4. (12 points)

Consider the following Markov decision process (MDP). It has states $\{0, 1, 2\}$. In every state, you can take one of two possible actions: A or B. State 0 is the terminal state (*i.e.*, once the agent reaches that state, it stays there no matter what action it takes).

The transition probabilities for action A and B are given as:

$T(s, A, s')$ $s \backslash s'$	0	1	2
0	1.0	0.0	0.0
1	0.5	0.5	0.0
2	0.5	0.0	0.5

$T(s, B, s')$ $s \backslash s'$	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0

For example, $T(2, A, 0) = 0.5$, $T(2, B, 1) = 1.0$.

The reward function is defined as $R(s, a, s') = s'^2$. The discount factor is $\gamma = 0.6$.

Let us consider the Q-value iteration algorithm.

1. Suppose we initialize $Q_0^*(s, a) = 0$ for all $s \in \{0, 1, 2\}$ and $a \in \{A, B\}$. Evaluate the Q-values $Q_1^*(s, a)$ after exactly one iteration of the Q-Value Iteration Algorithm. Write your answers in the table below.

	$s = 0$	$s = 1$	$s = 2$
A	0	0.5	2
B	0	0	1

0 0.8 3.2

0 0 1.3

2. What is the policy that we would derive from $Q_1^*(s, a)$? Answer by filling in the action that should be taken at each state in the table below.

$s = 1$	$s = 2$
A	A

3. What are the values $V_1^*(s)$ corresponding to $Q_1^*(s, a)$?

$s = 0$	$s = 1$	$s = 2$
0	0.5	2

4. Will the policy change after the second iteration? If your answer is "yes", briefly describe how.

No.