KOÇ UNIVERSITY
COLLEGE OF ENGINEERING

COMP 341: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Final

FALL 2016, 12/01/2017
DURATION: 150 MINUTES

Name: Solutions

ID: 00110001 00110000 00110000

- This exam contains 11 pages including this cover page and 8 questions. Check to see if any pages are missing. Put your initials on the top of every page, in case the pages become separated.
- By submitting this exam, you **agree** to fully comply with Koç University Student Code of Conduct, and accept any punishment in case of failure to comply. If you do not submit this exam on time, you will receive 0 credits.
- The exam is **closed book** and **closed notes**. You are **not** allowed to use any additional material including any electronic equipment such as computers and mobile phones.
- You are expected to be able provide clear and concise answers. Gibberish will not receive any credit.
- Your answers need to be readable by a human, illegible writing is not gradable hence there is a strong chance that such answers will not get any credit.
- Read each question carefully and make sure to follow instructions. The written instructions take precedence over an answer that the instructor might give you during the exam, unless the instructor makes a class wide announcement.
- Do not write in the table below.

Question:	1	2	3	4	5	6	7	8	Total
Points:	8	12	10	10	12	18	16	14	100
Score:									

1. (8 points) True or False :
   ML: Machine Learning
   kNN: K-Nearest Neighbors
   GMM: Gaussian Mixture Model
   NB: Naïve Bayes
   MDP: Markov Decision Process
   RL: Reinforcement Learning

   False In ML, one way to overcome underfitting issues is to get more data.

   False In the kNN method, increasing k would also increase the chance of overfitting.

   False The initialization does not matter for GMM learning.

   True In the NB method, we assume the input features are independent effects of the output.

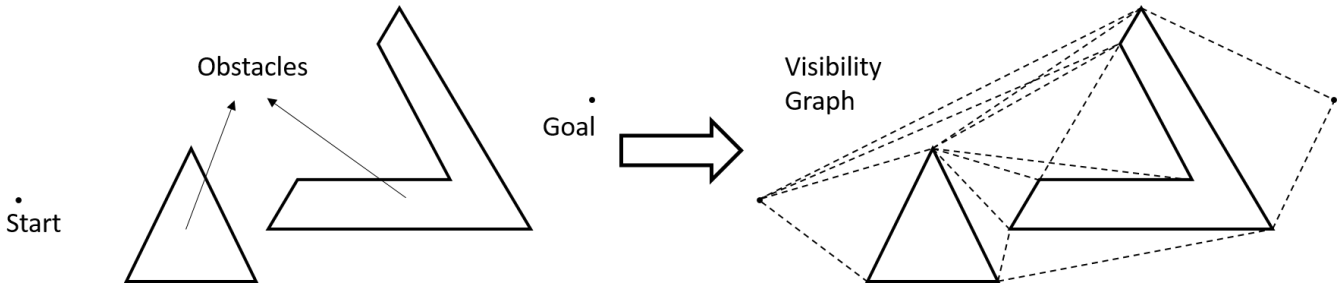
   True Expectiminimax algorithm can be modified to solve an MDP.

   True In MDPs, the discount factor is a part of the problem.

   False The transition model must be known to calculate Q-values in RL.

   True The methods we have seen for RL assume that states are directly observable.

2. (12 points) Search:
   You are doing an inter-disiplinary Comp491 project about a wheeled robot that needs to move in an environment with polygonal obstacles. Your ME and EE friends are developing the robot hardware, another COMP student is getting the obtacle shapes and locations, and robot position from an overhead vision system. You are developing the motion planning algorithm to move between two points. You decide to use *visibility graphs*. A visibility graph is an undirected graph. Its vertices are composed of the vertices of the polygonal obstacles, the start point of the robot and the goal point. Its edges are composed of all the edges of the polygonal objects, plus all the edges between vertices that can be connected together without crossing an obstacle. You read that the optimal path within this graph would give you the optimal path in the real world as well. An example and the resulting graph, where the dashed lines represent the edges that can be added between vertices without crossing an obstacle:



(The question continues on the next page)

(a) (8 points) Assume that you can generate the visibility graph and want the optimal path. Formulate the search problem. What are your states, transition function, cost, start state and the goal test? Which algorithm would you use to find the path within this graph and why? What would be a heuristic that you would use and why?

1 point:

States: The start point, goal point and the vertices of the obstacles
Transition Function: Move between the connected neighbors
Cost: Euclidean Distance
Start state: The start point
Goal Test: Check whether the current state is the goal point

1.5 points:

I would use A* algorithm since it is optimal and faster than UCS
My heuristic would be the Euc. Dist. to the goal point since it is a consistent heuristic.

(b) (4 points) Another way would have been to create a grid and mark the cells with an object as occupied. Then the problem becomes calculating a path between the cells that contain the start and the goal point. Comment on the state space sizes and the branching factor between the two approaches. Define any necessary parameters you might need.

Let v be the number of vertices in the visigiblity graph (VG). Let k be the number of average visible vertices from each vertex, i.e., the average number of neighbors in the VG

State space size for the grid case is n^2, for the VG case is v. We expect the VG case to be smaller. We expect for VG be more scalable for larger spaces and the algorithm to solve would be complete and optimal vs resolution complete and optimal for the grid case.

Branching factor for the grid case is 4 or 8, depending on the neighborhood selection and k for the VG. We expect k to be potentially larger than 4 and perhaps 8.

3. (10 points) First Order Logic: Given the logic knowledge base, try to generate $Murderer(Alfred)$ and $Murderer(Jack)$ using forward chaining, i.e., repeated application of the generalized modus ponens. You are free to select any pick order you want. Write the substitutions, existing facts and the resulting facts clearly in the provided tables. Leave the substitutions empty if the variable does not appear in the rule. The variables are per rule and not "global". The first one is done as an example for you. You can use the generated facts between the tables. The length of the tables do not indicate solution length.

Rules:
$R1 : \forall x \;\; Doctor(x) \Rightarrow Rich(x)$
$R2 : \forall x \forall y \;\; Angry(x) \wedge Butler(x) \wedge Boss(y, x) \Rightarrow Kill(x, y)$
$R3 : \forall x \forall y \forall z \;\; Gardner(x) \wedge Married(y, z) \wedge Greedy(y) \wedge Rich(z) \Rightarrow Kill(x, z)$
$R4 : \forall x \forall y \forall z \;\; InLove(x, y) \wedge Married(y, z) \wedge x \neq z \Rightarrow Angry(x)$
$R5 : \forall x \forall y \;\; Kill(x, y) \Rightarrow Murderer(x)$
$R6 : \forall x \forall y \;\; Married(x, y) \Rightarrow Married(y, x)$
Facts:

$F1 : Doctor(John)$	$F2 : Doctor(Jane)$	$F3 : Greedy(John)$
$F4 : Gardner(Jack)$	$F5 : Butler(Alfred)$	$F6 : Married(John, Jane)$
$F7 : InLove(Alfred, Jane)$	$F8 : Boss(John, Alfred)$	$F9 : Boss(John, Jack)$

$Murderer(Alfred)$:

Rule	x	y	z	Used Facts	New Fact
$R6$	$John$	$Jane$		$F6$	$F10 : Married(Jane, John)$
$R4$	$Alfred$	$Jane$	$John$	$F7, F10$	$F11 : Angry(Alfred)$
$R2$	$Alfred$	$John$		$F11, F5, F8$	$F12 : Kill(Alfred, John)$
$R5$	$Alfred$	$John$		$F12$	$F13 : Murderer(Alfred)$

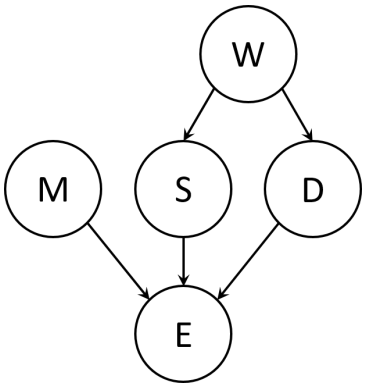
$Murderer(Jack)$:

Rule	x	y	z	Used Facts	New Fact
$R1$	$Jane$			$F2$	$F14 : Rich(Jane)$
$R3$	$Jack$	$John$	$Jane$	$F4, F6, F3, F14$	$F15 : Kill(Jack, Jane)$
$R5$	$Jack$	$Jane$		$F15$	$F16 : Murderer(Jack)$

2 points per $R2$, $R3$ and $R4$ lines, 1 point for others, 1 points for clarity

4. (10 points) Bayesian Networks:
You like to play real time strategy games. Based on your experience you built a BN. You want to infer whether your opponent is likely to do an early rush based on your scouting. Your variables are:

- **Map size (M):** Whether the map is small (+) or large (−)
- **Workers (W):** Whether your opponent is concentrating on building workers (+) or not (−)
- **Soldiers (S):** Whether your opponent is concentrating on building soldiers (+) or not (−)
- **Defence (D):** Whether your opponent is building defence structures (+) or not (−)
- **Early rush (E):** Whether your opponent is going to early rush (+) or not (−)



(Note that you do not need to know about RTS games to solve this problem.)

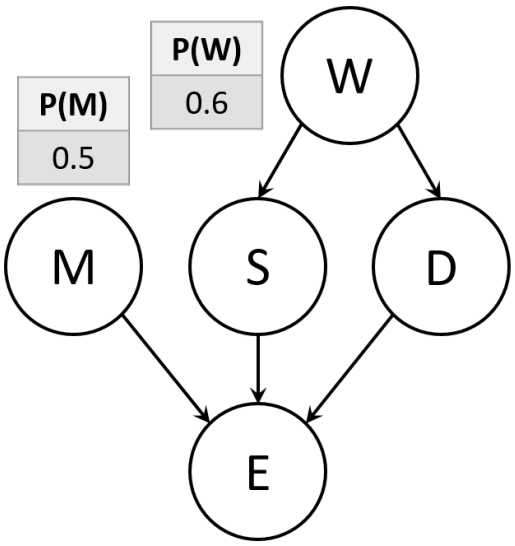
(a) (2 points) Your opponent tried to early rush you in a small map with a large army but failed. You want to go on the offensive and want to infer if he has any defensive structures, i.e., $P(+d|+m,+s,+e)$. Write down the expression(s) for the **factor(s)** to calculate this with variable elimination.
Current factors are: $P(+m), P(W), P(+s|W), P(+d|W), P(+e|+m,+s,+d)$
We only have W as the free varible so we sum it out. You can either calculate 1 factor or calculate multiple factors.
$f_1(+d,+s) = \sum_{w=\{-w,+w\}} P(w)P(+d|w)P(+s|w)$, $f_2(+d,+s,+m,+e) = f_1(+d,+s)P(+e|+m,+s,+d)$
OR
$f(+d,+m,+s,+e) = P(+e|+m,+s,+d)\sum_{w=\{-w,+w\}} P(w)P(+d|w)P(+s|w)$
Then you get the $-d$ versions and normalize.

(b) (2 points) Suddenly you do not want to do variable elimination anymore and decide to use rejection sampling instead. Using the samples below, calculate the likelihood of your opponent doing an early rush in a small map $P(+e|+m)$

M	W	S	D	E
-	-	+	+	+
-	+	-	+	-
+	+	+	-	+
-	+	-	-	-
+	-	+	-	+
+	-	+	-	+
-	-	+	-	-
+	+	-	+	-
+	+	+	-	+
+	-	+	-	+

Number of $+m$ is 6 and number of $+m,+e$ together is 5. Then, $P(+e|+m) = 5/6$

(c) (2 points) Based on the rejection sampling using the samples above, estimate the likelihood of your opponent ammassing an army if he did not do an early rush and you observed him building workers, i.e., $P(+s|+w,-e)$
Number of $+w,-e$ together is 3 and number of $+s,+w,-e$ together is 0. Then, $P(+s|+w,-e) = 0/3 = 0$



W	P(S|W)
+w	0.2
-w	0.7

W	P(D|W)
+w	0.4
-w	0.3

M	S	D	P(E|M,S,D)
+m	+s	+d	0.4
+m	+s	-d	0.9
+m	-s	+d	0.2
+m	-s	-d	0.5
-m	+s	+d	0.3
-m	+s	-d	0.7
-m	-s	+d	0.1
-m	-s	-d	0.2

(d) (2 points) You realized you have rejected many samples for the previous query $P(+s|+w,-e)$ and want to be more data efficient. You draw the following samples, using likelihood weighting. The necessary probabilities, for the true cases, are given above. Calculate the weights of the samples given below.

M	W	S	D	E	Weight
+	+	+	+	-	$0.6 \times 0.6 = 0.36$
-	+	-	-	-	$0.6 \times 0.8 = 0.48$
-	+	-	-	-	$0.6 \times 0.8 = 0.48$
+	+	-	-	-	$0.6 \times 0.5 = 0.30$

(e) (2 points) Based on the samples from the likelihood weigthing, estimate $P(+s|+w,-e)$.
$P(+s|+w,-e) = 0.36/(0.36+0.48+0.48+0.3) = 0.36/1.62 = 2/9 = 0.2222$

5. (12 Points) Decision Trees
You are given a set of labeled training examples below, where each feature ($F1$ and $F2$) has three possible values a, b, or c. You choose to learn a decision tree and select "-" as the default output if there are ties.

$F1$	$F2$	$Output$
c	a	+
b	c	+
a	b	+
a	a	-
c	b	-
b	b	-

(a) (9 points) What score would the information gain calculation assign to each of the features? Leave any logarithms as is but simplify as much as you can. Show all your work.
Let us denote the number of positive and negative examples as $\{p, n\}$
Let $B(q)$ be the entropy for the boolean output where $q = p/(p+n)$ is the ratio of positives to the rest. Then $B(q) = -qlog_2(q) - (1-q)log_2(1-q)$
Also let $q_i = p_i/(p_i+n_i)$ denote the ratio of the positives to the rest for the i^{th} value of a feature with p_i positive and n_i negative examples.
Let the remainder $R(F)$ denote the total entropy for all the splits if the feature F is selected. $R(F) = \sum_i \frac{p_i+n_i}{p+n} B(q_i)$
Then the gain when selecting F is $G(F) = B(q) - R(F)$

Before the split we have equal number of positives and negatives as $\{3, 3\}$:
$B(0.5) = -0.5log_2(0.5) - (1-0.5)log_2(0.5) = 1$

For F1 all the values have the same number of positives and negatives as $\{1, 1\}$, which all have the same $B(q) = 1$
We can calculate the remainder as: $R(F1) = 3(\frac{2}{6} \times 1) = 1$
Then the gain is G(F1) = 1 - 1 = 0

For F2, we have different values $q_a = 1/2, q_b = 1/3, q_c = 1$
$B(q_a) = B(0.5) = 1$ from above
$B(q_c) = B(1) = -1log_2(1) - (1-1)log_2(0) = 0$
$B(q_b) = B(1/3) = -1/3log_2(1/3) - (1-1/3)log_2(1-1/3) =$
$-1/3(log_2(1) - log_2(3)) - 2/3(log_2(2) - log_2(3)) = log_2(3) - 2/3$
We can calculate the remainder as: $R(F2) = \frac{2}{6} \times 1 + \frac{3}{6} \times (log_2(3) - 2/3) + \frac{1}{6} \times 0 = log_2(3)/2$
Then the gain is $G(F2) = 1 - log_2(3)/2 > 0$

Points: +1 for calculating simple things (e.g. B(0.5), B(1), R(F1),R(F2), G(F1),G(F2)), +2 for calculating more complicated entropy (e.g. B(2/3)), +1 for simplifying

(b) (3 points) Which feature would be chosen as the root of the decision tree?
F2, because its information gain is higher

6. (18 Points) Consider the following set of 6 points in the plane:
$P = \{A = (0;0); B = (1;0); C = (0;3); D = (1;3); E = (5;7); F = (8;12)\}$

You'd like to partition the points into two clusters, where each cluster is represented by a centroid μ_k that is constrained to lie on the diagonal line, i.e. , $\mu_k = (c_k; c_k)$ for $k \in \{1, 2\}$. Modify the K-means algorithm to minimize the squared sum of distances (denoted by J) to the centroids while respecting the diagonal constraint. Answer the following questions.

(a) (6 points) Derive the modified equation to update c_k's given the constraint, $\mu_k = (c_k; c_k)$. (Hint: Start by writing what J is)

$J = \sum_{k=1}^{2} \sum_{x \in P_k} (c_k - x_1)^2 + (c_k - x_2)^2$, where x denotes the points in the given cluster, k, and its subscripts denote the dimension

$\frac{\delta J}{\delta c_k} = \sum_{x \in P_k} 2(c_k - x_1) + 2(c_k - x_2) = 2n_k c_k - \sum_{x \in P_k} (x_1 + x_2)$, where n_k is the number of points in cluster k

$\frac{\delta J}{\delta c_k} = 0 \Rightarrow c_k = \frac{\sum_{x \in P_k} (x_1 + x_2)}{2n_k}$

(b) (12 points) Suppose the 2 clusters are initialized with $c_1 = 3$ and $c_2 = 11$. Run two iterations of K-means, filling out the table below. Break ties, if any, by assigning a point to the centroid farther away from the origin. You can use the back of this sheet for calculations.

Iter.	Task	Value
1	Assignment $z_i = 1$	A, B, C, D, E
1	Assignment $z_i = 2$	F
1	New c_1	2
1	New c_2	10
2	Assignment $z_i = 1$	A, B, C, D
2	Assignment $z_i = 2$	E, F
2	New c_1	1
2	New c_2	8

7. (16 points) Markov Decision Processes:
Consider a server that has to serve two queues denoted by a and b. Each queue can hold at most 5 requests. At a given time, the server choses between continuing to serve the current queue it is serving or to switching to the other queue. In case it chooses to serve, a request is removed from the current queue. Serving takes an entire time step. Switching also takes an entire time step so no queue is served in that time step. In addition, at each time step, for each queue that is shorter than five, a request is added to that queue with probability p_a and p_b respectively. This happens independently for each queue. During a time step, the requests are added after the queue is served. Serving a request results in a reward of +1. Formulate an MDP, (S, A, T, γ, R), that models this setting.
2 points for S, 1 point for A, 1 point for γ, 9 points for T and 3 points for R
$S = (n_a, n_b, q)$, where $0 \leq n_a, n_b \leq 5$ denote the current number of requests in the queues and q is the current queue
$A = \{serve, switch\}, \gamma \in [0, 1]$
$T(s, a, s')$ or $P(s'|s, a)$.
Let $a = serve$:
The symmetric can be easily found by changing subscripts, if the current queue was b

current state	condition	next state	probability
(n_a, n_b, a)	$0 < n_a \leq 5, 0 \leq n_b < 5$	$(n_a - 1, n_b, a)$	$(1 - p_a)(1 - p_b)$
		(n_a, n_b, a)	$(p_a)(1 - p_b)$
		$(n_a - 1, n_b + 1, a)$	$(1 - p_a)p_b$
		$(n_a, n_b + 1, a)$	$(p_a)(p_b)$
	$n_a = 0, 0 \leq n_b < 5$	$(0, n_b, a)$	$(1 - p_b)$
		$(0, n_b + 1, a)$	p_b
	$0 < n_a \leq 5, n_b = 5$	$(n_a - 1, 5, a)$	$(1 - p_a)$
		$(n_a, 5, a)$	p_a
	$n_a = 0, n_b = 5$	$(0, 5, a)$	1

Let $a = switch$:
The symmetric can be easily found by changing the queue names between the current and the next state, if the current queue was b

current state	condition	next state	probability
(n_a, n_b, a)	$0 \leq n_a, n_b < 5$	(n_a, n_b, b)	$(1 - p_a)(1 - p_b)$
		$(n_a + 1, n_b, b)$	$(p_a)(1 - p_b)$
		$(n_a, n_b + 1, b)$	$(1 - p_a)p_b$
		$(n_a + 1, n_b + 1, b)$	$(p_a)(p_b)$
	$n_a = 5, 0 \leq n_b < 5$	$(5, n_b, b)$	$(1 - p_b)$
		$(5, n_b + 1, b)$	p_b
	$0 \leq n_a < 5, n_b = 5$	$(n_a, 5, a)$	$(1 - p_a)$
		$(n_a + 1, 5, b)$	p_a
	$n_a = 5, n_b = 5$	$(5, 5, b)$	1

$R((n_a, n_b, a), serve) = 1$ if $n_a > 0$, $R((n_a, n_b, b), serve) = 1$ if $n_b > 0$, $R(\cdot) = 0$, otherwise

8. (14 points) Reinforcement Learning:
You are helping a tribe of "cavemen" to survive. The cavemen are simple people. They $hunt(a_h)$, $eat(a_e)$ or $sleep(a_s)$ and only have a few states related to survival; $hungry(s_h)$, $full(s_f)$, $gotfood(s_g)$ and $dead(s_d)$. Note that dead is a terminal state. They like to live in the moment more than caring too much about the future, so you set the discount factor at 0.8. You want to come up with an optimal policy for their survival. Since you are not sure about their transition and reward functions you decide to use Q-learning (you quantify reward based on how they look after a transition). You observe one of them and note the following state-action-reward sequence.

s	a	s	r	a	s	r	a	s	r	a	s	r	a	s	r	a	s	r
s_h	a_h	s_g	1	a_e	s_f	8	a_s	s_h	0	a_s	s_h	0	a_h	s_g	1	a_e	s_h	0

(a) (9 points) With $\alpha = 0.5$, run Q-learning on the given sequence with $Q_0 = 0$ for all states and fill the following table. Carry the previous values for the unaffected Q-values between transitions. For partial credit, show your work clearly at the spaces after the table and the back of this sheet. Leave any mathematical expressions as is if you are not able to compute them.

Q	Transition 1	Transition 2	Transition 3	Transition 4	Transition 5	Transition 6
(s_h, a_e)	0	0	0	0	0	0
(s_h, a_h)	0.5	0.5	0.5	0.5	2.35	2.35
(s_h, a_s)	0	0	0	0.2	0.2	0.2
(s_f, a_e)	0	0	0	0	0	0
(s_f, a_h)	0	0	0	0	0	0
(s_f, a_s)	0	0	0.2	0.2	0.2	0.2
(s_g, a_e)	0	4	4	4	4	2.94
(s_g, a_h)	0	0	0	0	0	0
(s_g, a_s)	0	0	0	0	0	0
$(s_d, \cdot)$	0	0	0	0	0	0

1.5 points per correct entry, -3 for wrong close enough update, -6 for completely wrong update
$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s,a,s') + \gamma max_{a'}(Q(s',a')) - Q(s,a))$
Transition 1: $Q(s_h, a_h) = 0 + 0.5(1 + 0.8 \cdot 0 - 0) = 0.5$
Transition 2: $Q(s_g, a_e) = 0 + 0.5(8 + 0.8 \cdot 0 - 0) = 4.0$
Transition 3: $Q(s_f, a_s) = 0 + 0.5(0 + 0.8 \cdot 0.5 - 0) = 0.2$
Transition 4: $Q(s_h, a_s) = 0 + 0.5(0 + 0.8 \cdot 0.5 - 0) = 0.2$
Transition 5: $Q(s_h, a_h) = 0.5 + 0.5(1 + 0.8 \cdot 4.0 - 0.5) = 2.35$
Transition 6: $Q(s_g, a_e) = 4.0 + 0.5(0 + 0.8 \cdot 2.35 - 4.0) = 2.94$

(b) (2 points) What is the optimal action for state $hungry(s_h)$? What is the optimal action for state $gotfood(s_g)$?
$\pi(s_h) = a_h$, $\pi(s_g) = a_e$

(c) (3 points) At the end of the last sequence, the caveman you were watching was still hungry so he decided to hunt again but he had an unfortunate accident and died and you estimated his reward as -10, i.e., you observed $(s_h, a_h, s_d, -10)$. Update $Q(s_h, a_h)$, using its previous estimate. What is the optimal action for state s_h now? Can you comment on this situation?
$Q(s_h, a_h) = 2.35 + 0.5(-10 + 0.8 \cdot 0 - 2.35) = -3.825$
$\pi(s_h) = a_s$
This is not good as he would die if he keeps sleeping while he is hungry. We hope that this was a rare event and to be able to come up with a better policy, we need to observe more cavemen.