CS 188 Fall 2012 — Introduction to Artificial Intelligence — Final

- You have approximately 3 hours.
- The exam is closed book, closed notes except your three one-page crib sheets.
- Please use non-programmable calculators only.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.

First name	
Last name	
SID	

First and last name of student to your left	
First and last name of student to your right	

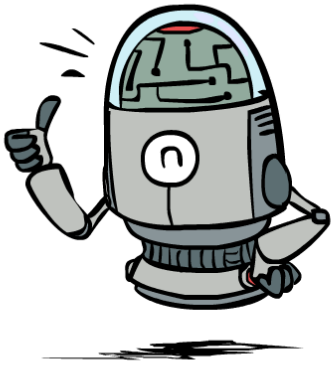
For staff use only:

Q1.	Warm-Up	/1
Q2.	Search: Return of the Slugs	/10
Q3.	CSPs: Enforcing Arc Consistency	/6
Q4.	CSPs and Bayes' Nets	/10
Q5.	Games: 3-Player Pruning	/18
Q6.	Reinforcement Learning	/6
Q7.	Bandits	/22
Q8.	Bayes' Nets Representation	/12
Q9.	Preprocessing Bayes' Net Graphs for Inference	/11
Q10.	Clustering	/8
Q11.	Decision Trees	/6
Q12.	Machine Learning: Overfitting	/10
	Total	/120

THIS PAGE IS INTENTIONALLY LEFT BLANK

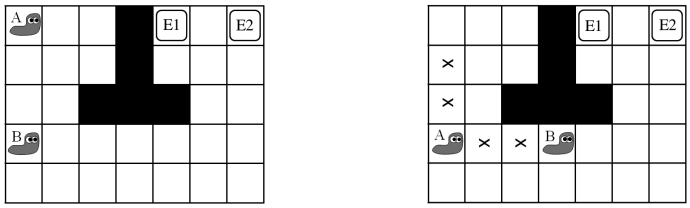
Q1. [1 pt] Warm-Up

Circle the CS188 mascot



Q2. [10 pts] Search: Return of the Slugs

As shown in the diagram on the left, two slugs A and B want to exit a maze via exits E_1 and E_2. At each time step, each slug can either stay in place or move to an adjacent free square. A slug cannot move into a square that the other slug is moving into. Either slug may use either exit, but they cannot both use the same exit. When a slug moves, it leaves behind a poisonous substance. The substance remains in the square for *2 time steps*; during this time, no slug can move into the poisonous square.



For example, if slugs A and B begin in the positions shown above on the left, and slug A moves *Down* 3 steps while slug B moves *Right* three steps, then the world becomes as shown above on the right, with $\times$'s marking the poisonous squares. Note that slug A *does* have the option of moving *Right* from its current position, since the trail from B will evaporate by the time it arrives.

You must pose a search problem that will get both slugs to the exits in as few time steps as possible. Assume that the board is of size M by N. While all answers should hold for a general board, not just the one shown above, you do *not* need to generalize beyond two slugs, two exits, or two timesteps of decay for the slug trails.

(a) [4 pts] How many states are there in a minimal representation of the space? Justify with a brief description of the components of your state space.

(b) [6 pts] Let $d(x, y)$ denote Manhattan distance between x and and y. Consider three heuristics, defined in terms of slug locations A and B and exit locations E_1 and E_2. Remember that either slug can use either exit, but they must use different exits.

Definition	*Explanation*
$h_1 = \max_{s \in \{A,B\}} \min(d(s, E_1), d(s, E_2))$	Return the maximum, over the two slugs, of the distance from the slug to its closest exit.
$h_2 = \max(d(A, E_1), d(B, E_2))$	Assign slug A to exit E_1 and B to E_2; then return the maximum distance from either slug to its assigned exit.
$h_3 = \min_{(e,e') \in \{(E_1,E_2),(E_2,E_1)\}} \max(d(A, e), d(B, e'))$	Return the max distance from a slug to its assigned exit, under the assignment of slugs to distinct exits which minimizes this quantity.

(i) [3 pts] Which of these three heuristics are admissible? Fill in all that apply.

◯ h_1 ◯ h_2 ◯ h_3

(ii) [3 pts] For each pair of heuristics, fill in the circle which correctly describes their dominance relationship. *Note: dominance is defined regardless of admissibility.*

◯ h_1 dominates h_2 ◯ h_2 dominates h_1 ◯ $h_1 = h_2$ ◯ none

◯ h_1 dominates h_3 ◯ h_3 dominates h_1 ◯ $h_1 = h_3$ ◯ none

◯ h_2 dominates h_3 ◯ h_3 dominates h_2 ◯ $h_2 = h_3$ ◯ none

Q3. [6 pts] CSPs: Enforcing Arc Consistency

You are given a CSP with three variables: A, B, and C. Each variable has an initial domain of $\{1, 2, 3\}$. The following constraints apply:

- $A \neq B$
- $A \neq C$
- $B > C$

(a) [4 pts] **Arcs:** You are asked to enforce arc-consistency in this CSP. Assume that the algorithm for enforcing arc consistency initially starts with *all* arcs in a queue in alphabetical order (as below). The queue is thereafter processed in a first-in-first-out manner, so any new arcs are processed in the order they are added.

Fill in the circles for any arcs that are *re-inserted* into the queue at some point. *Note: if an arc is already in the queue, it cannot be "added" to the queue.*

◯ $A \rightarrow B$

◯ $A \rightarrow C$

◯ $B \rightarrow A$

◯ $B \rightarrow C$

◯ $C \rightarrow A$

◯ $C \rightarrow B$

(b) [2 pts] **Domains:** Cross off the values that are eliminated from each variable's domain below:

A	1	2	3
B	1	2	3
C	1	2	3

Q4. [10 pts] CSPs and Bayes' Nets

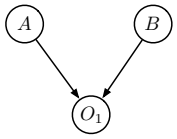
We consider how to solve CSPs by converting them to Bayes' nets.

(a) A Simple CSP

First, assume that you are given a CSP over two binary variables, A and B, with only one constraint: $A \neq B$.

As shown below, the Bayes' net is composed of nodes A and B, and an additional node for a binary random variable O_1 that represents the constraint. $O_1 = +o_1$ when A and B satisfy the constraint, and $O_1 = -o_1$ when they do not. The way you will solve for values of A and B that satisfy the CSP is by running inference for the query $P(A, B \mid +o_1)$. The setting(s) of A, B with the highest probability will satisfy the constraints.

(i) [2 pts] **Fill in the values** in the rightmost CPT that will allow you to perform the inference query $P(A, B \mid +o_1)$ with correct results.



A	$P(A)$
$-a$	0.5
$+a$	0.5

B	$P(B)$
$-b$	0.5
$+b$	0.5

A	B	O_1	$P(O_1 \mid A, B)$
$-a$	$-b$	$-o_1$	
$-a$	$-b$	$+o_1$	
$-a$	$+b$	$-o_1$	
$-a$	$+b$	$+o_1$	
$+a$	$-b$	$-o_1$	
$+a$	$-b$	$+o_1$	
$+a$	$+b$	$-o_1$	
$+a$	$+b$	$+o_1$	

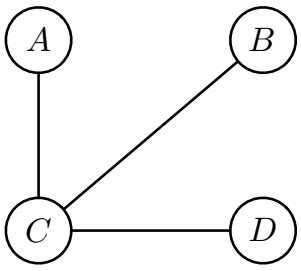
(ii) [2 pts] **Compute the Posterior Distribution:** Now you can find the solution(s) using probabilistic inference. Fill in the values below.

O_1	A	B	$P(A, B \mid +o_1)$
$+o_1$	$-a$	$-b$	
$+o_1$	$-a$	$+b$	
$+o_1$	$+a$	$-b$	
$+o_1$	$+a$	$+b$	

(b) [3 pts] **A More Complex CSP**

The graph below on the left shows a more complex CSP, over variables A, B, C, D. Recall that nodes in this graph correspond to variables that can be assigned values, and an edge between two nodes corresponds to a constraint that involves both of the variables.

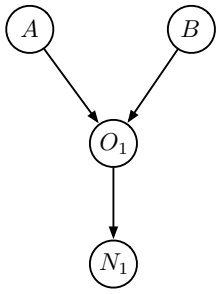
To the right of the graph below, **draw the Bayes' Net** that, when queried for the posterior $P(A, B, C, D \mid +o_1, +o_2, +o_3)$, will assign the highest probability to the setting(s) of values that satisfy all constraints. You do not need to specify any of the CPTs.



(c) [3 pts] **Unsatisfiable Constraints**

It may be the case that there is no solution that satisfies all constraints of a CSP. We decide to deal with this by adding an extra set of variables to our Bayes' Net representation of the CSP: for each constraint variable O_i, we add a child binary random variable N_i for constraint satisfaction, and condition on $+n_i$ instead of on $+o_i$.

For example, the Bayes' Net for the two-variable CSP from before will look like the graph on the right. Furthermore, we construct the new CPT to have the form of the CPT on the right.



O_1	N_1	$P(N_1 \mid O_1)$
$-o_1$	$-n_1$	p
$-o_1$	$+n_1$	$1-p$
$+o_1$	$-n_1$	q
$+o_1$	$+n_1$	$1-q$

Consider a CSP over three binary variables A, B, C with the constraints that no two variables have the same value.

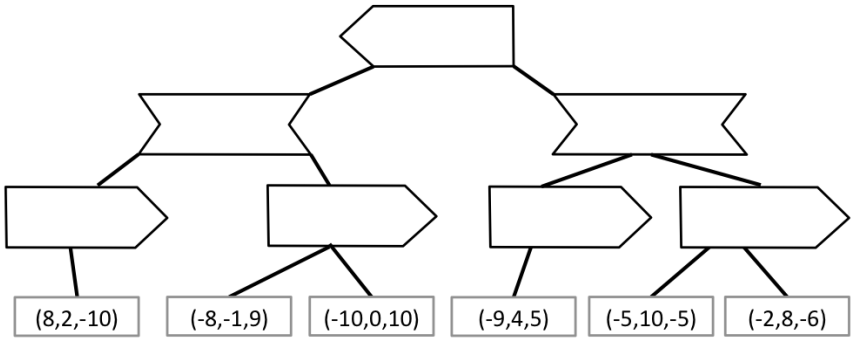
Write an expression below in terms of p and/or q, and standard mathematical operators (e.g. $pq < 0.1$ or $p = q$), that will guarantee that solutions satisfying more constraints have higher posterior probability $P(A, B, C \mid +n_1, +n_2, +n_3)$. Keep in mind that not all constraints can be satisfied at the same time.

Your expression:

Q5. [18 pts] Games: 3-Player Pruning

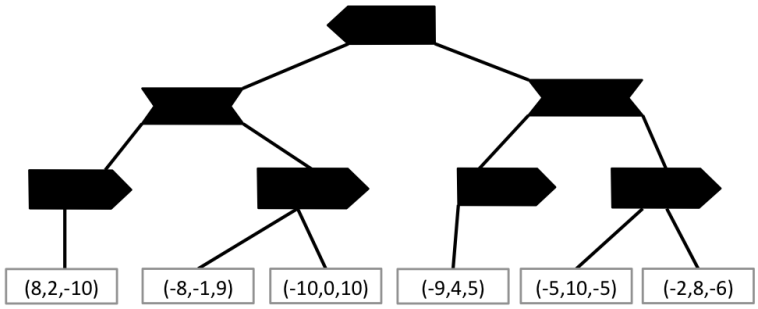
(a) [2 pts] **A 3-Player Game Tree**

Consider the 3-player game shown below. The player going first (at the top of the tree) is the Left player, the player going second is the Middle player, and the player going last is the Right player, optimizing the left, middle and right components respectively of the utility vectors shown. Fill in the values at all nodes. Note that all players *maximize* their own respective utilities.



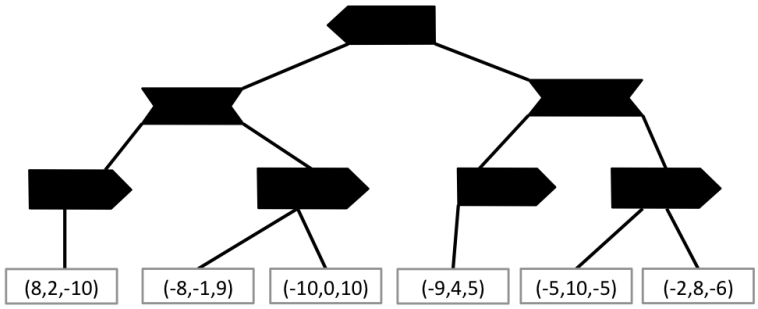
(b) [3 pts] **Pruning for a 3-Player Zero-Sum Game Tree**

We would like to prune nodes in a fashion similar to α - β pruning. Assume that we have the knowledge that the sum of the utilities of all 3 players is always zero. What pruning is possible under this assumption? Below, cross off with an $\times$ any branches that can be safely pruned. If no branches can be pruned, justify why not:



(c) [3 pts] **Pruning for a 3-Player Zero-Sum, Bounded-Utility Game Tree.**

If we assume more about a game, additional pruning may become possible. Now, in addition to assuming that the sum of the utilities of all 3 players is still zero, we also assume that all utilities are in the interval $[-10, 10]$. What pruning is possible under these assumptions? Below, cross off with an $\times$ any branches that can be safely pruned. If no branches can be pruned, justify why not:



(d) [6 pts] **Pruning Criteria**

Again consider the zero-sum, bounded-utility, 3-player game described above in (c). Here we assume the utilities are bounded by $[-C, C]$ for some constant C (the previous problem considered the case $C = 10$). Assume there are only 3 actions in the game, and the order is Left, Middle, then Right. Below is code that computes the utilities at each node, but the pruning conditions are left out. Fill in conditions below that prune wherever safe.

(i) [2 pts] **Pruning for Left**

```
Compute-Left-Value(state, C)
  v_L = -infty; alpha = -infty
  for each successor of state:
    (w_L, w_M, w_R) = Compute-Middle-Value(successor, C, alpha)
    if (w_L > v_L)
      v_L = w_L
      v_M = w_M
      v_R = w_R
    end
    // pruning condition:
    if (                  )
      return (v_L, v_M, v_R)
    end
    alpha = max(alpha, v_L)
  end
  return (v_L, v_M, v_R)
```

Fill in the pruning condition below, write $false$ if no pruning is possible:

( )

(ii) [2 pts] **Pruning for Middle**

```
Compute-Middle-Value(state, C, alpha)
  v_M = -infty; beta = -infty
  for each successor of state:
    (w_L, w_M, w_R) = Compute-Right-Value(successor, C, beta)
    if (w_M > v_M)
      v_L = w_L
      v_M = w_M
      v_R = w_R
    end
    // pruning condition:
    if (                  )
      return (v_L, v_M, v_R)
    end
    beta = max(beta, v_M)
  end
  return (v_L, v_M, v_R)
```

Fill in the pruning condition below, write $false$ if no pruning is possible:

( )

(iii) [2 pts] **Pruning for Right**

```
Compute-Right-Value(state, C, beta)
  v_R = -infty;  gamma = -infty
  for each successor of state:
    (w_L, w_M, w_R) = Evaluate-Utility(successor)
    if (w_R > v_R)
      v_L = w_L
      v_M = w_M
      v_R = w_R
    end
    // pruning condition:
    if (                  )
      return (v_L, v_M, v_R)
    end
    gamma = max(gamma, v_R)
  end
  return (v_L, v_M, v_R)
```

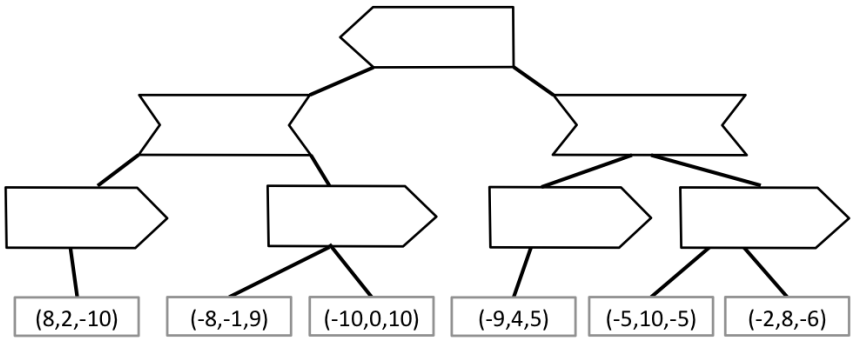
Fill in the pruning condition below, write $false$ if no pruning is possible:

( )

(e) [4 pts] **Preparing for the Worst (Case)**

Consider again the game tree from earlier, shown again below. Assume it is not known whether Middle and Right are rational. Left wants to avoid the worst-case scenario according to its own utilities.

Note: you do not need to fill in this game tree; it is provided purely to aid you in answering the question below.



(i) [2 pts] What is Left's optimal action? What is the utility achieved in this worst-case scenario?

(ii) [2 pts] Which one of the following approaches would allow, for a generic 3-player game, the Left player to compute the strategy that optimizes its worst-case outcome? (you may ignore pruning here)

◯ Running the strategy described in the pseudo-code from question (d).

◯ Running standard minimax, considering Left's utility only, considering Left to be a maximizer and considering both Middle and Right to be minimizers.

◯ Running standard expectimax, considering Left's utility only, considering Left to be a maximizer and considering both Middle and Right to be chance nodes.

◯ None of the above.

Q6. [6 pts] Reinforcement Learning

Recall that reinforcement learning agents gather tuples of the form $< s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1} >$ to update the value or Q-value function. In both of the following cases, the agent acts at each step as follows: with probability 0.5 it follows a fixed (not necessarily optimal) policy π and otherwise it chooses an action uniformly at random. Assume that in both cases updates are applied infinitely often, state-action pairs are all visited infinitely often, the discount factor satisfies $0 < \gamma < 1$, and learning rates α are all decreased at an appropriate pace.

(a) [3 pts] The Q-learning agent performs the following update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Will this process converge to the optimal Q-value function? If yes, write "Yes." If not, give an interpretation (in terms of kind of value, optimality, etc.) of what it will converge to, or state that it will not converge:

(b) [3 pts] Another reinforcement learning algorithm is called SARSA, and it performs the update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Will this process converge to the optimal Q-value function? If yes, write "Yes." If not, give an interpretation (in terms of kind of value, optimality, etc.) of what it will converge to, or state that it will not converge:

Q7. [22 pts] Bandits

You are in charge of sending monthly supply shipments to a remote research base, located deep in the jungle. The shipments can go by *Air* or by *Land*: a successful land shipment yields reward 10, while air shipments are smaller and yield reward 5. Air shipments always arrive safely. However, shipping over land is risky: the jungle is a lawless place and shipments may be vulnerable to attacks by roving bandits! You don't know whether there are bandits in this particular stretch of jungle; if there are, then each land shipment has only a 1/10 chance of arriving safely. If there are no bandits, then each land shipment has an 8/10 chance of arriving safely. You are in radio contact with the base, so you observe whether each month's shipment arrives or is lost.

State	Action	Obs	$P(o\|s,a)$
$+Bandits$	Air	$+Safe$	1
$+Bandits$	Air	$-Safe$	0
$+Bandits$	$Land$	$+Safe$	0.1
$+Bandits$	$Land$	$-Safe$	0.9
$-Bandits$	Air	$+Safe$	1
$-Bandits$	Air	$-Safe$	0
$-Bandits$	$Land$	$+Safe$	0.8
$-Bandits$	$Land$	$-Safe$	0.2

Table 1: Probability of safe arrival, for each state and action.

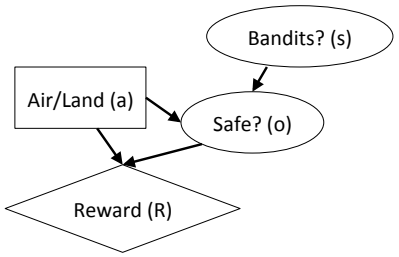


Figure 1: Decision diagram for a single timestep.

State	$P(s)$
$+Bandits$	0.5
$-Bandits$	0.5

(a) Prior probabilities

Action	Obs	$R(a,o)$
Air	$+Safe$	5
Air	$-Safe$	0
$Land$	$+Safe$	10
$Land$	$-Safe$	0

(b) Reward payoffs

Table 2

(a) [7 pts] **Value of information.** First consider the decision problem for a single timestep, shown in the decision diagram above. An explorer has emerged from the jungle and is offering to sell you the information of whether there are bandits in this particular stretch of jungle. How much should you pay? Given a uniform prior belief as to the presence of bandits (as indicated in Table 2a above), compute each of the following quantities to find the value of perfect information:

$MEU(\{\}) =$

$MEU(\{+Bandits\}) =$

$MEU(\{-Bandits\}) =$

$VPI(Bandits) =$

(b) Reasoning in belief space. Now we'll consider making sequential decisions over time.

We would like to formulate this problem as an MDP. We can model this problem as a *partially observed* MDP (POMDP), with two states $+Bandits$ and $-Bandits$ indicating the presence and lack of bandits respectively, and two actions *Air* and *Land* for sending the shipment by air or by land respectively. Note that the actual state of this MDP never changes – either there are bandits or there aren't – so the transition function of this MDP is just the identity. At each timestep we receive an observation of either $+Safe$ or $-Safe$, indicating respectively that the shipment arrived safely or was lost. The observation and reward models are given in Tables 1 and 2b respectively.

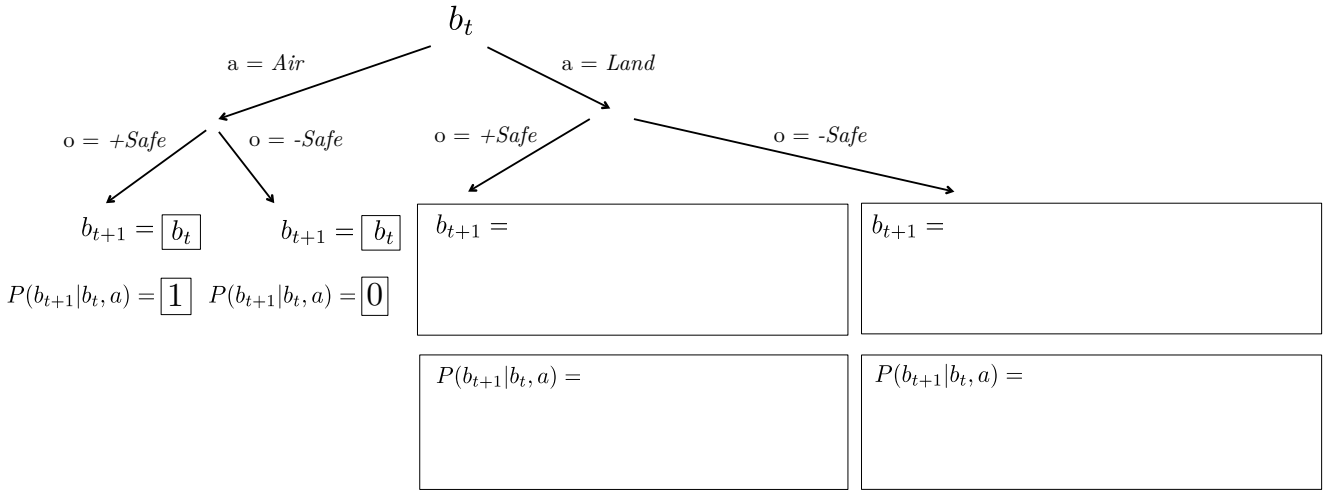
One approach to solving POMDPs is to work in terms of the equivalent *belief MDP*, a new (fully observed) MDP whose state space is the space of *distributions* over the states of the real-world MDP. In our case, each distribution is specified by a real number b giving the probability that there are bandits in the jungle, so the state space of the belief MDP is the continuous interval $[0, 1]$. Our uniform prior belief corresponds to a start state of $b_0 = 0.5$ in the belief MDP.

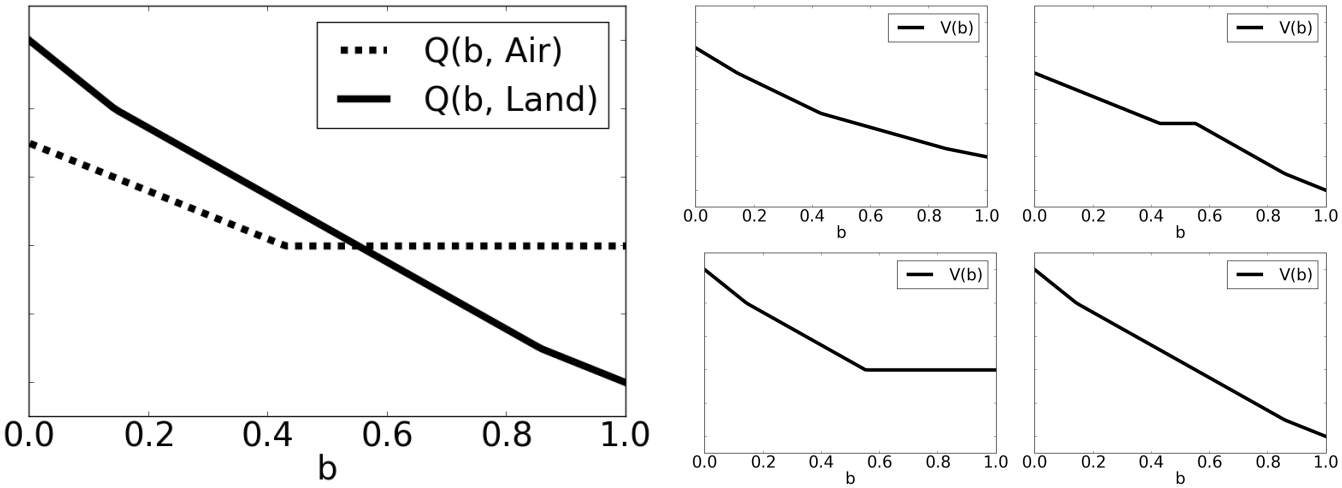
(i) [2 pts] Suppose we send the first shipment by land. Based on our starting belief state, what is the probability the shipment arrives safely?

(ii) [2 pts] After shipping by land we will observe either $o_1 = +Safe$ or $o_1 =$ -*Safe*, and based on this observation will transition to a new belief state $b_1 = P(s = +Bandits|o_1)$. Give the updated belief state b_1 for each outcome:

Outcome	b_1
$o_1 = +Safe$	
$o_1 =$ -*Safe*	

(iii) [4 pts] Give the general transition function $P(b_{t+1}|b_t, a)$ for the belief MDP. That is, in the outcome tree below, label each leaf node in the tree below with an expression for the updated belief b_{t+1} at that node in terms of an original belief b_t, and also give the probability of reaching that node. Note that an air shipment provides no information and so doesn't change your beliefs, so we have labeled those branches for you as an example.





(a) (b)

Figure 2

(c) [2 pts] **Value functions**. Suppose that Figure 2a shows, for some discount factor γ, the optimal Q-values for both actions plotted as a function of the belief b. Which of the plots in Figure 2b gives the value function $V(b)$?

◯ Top left ◯ Top right ◯ Bottom left ◯ Bottom right

(d) Discounting.

(i) [2 pts] You observe two other agents A and B acting in this belief MDP. All you see are their action sequences:

Agent A: $Land, Land, Air, Air, Air, Air, \ldots$ (followed by Air forever)
Agent B: $Land, Air, Air, Air, Air, Air, \ldots$ (followed by Air forever)

Supposing that these are rational agents, planning with an infinite time horizon and using the reward model from Table 2b, what if anything can you conclude about their discount factors γ_A and γ_B?

◯ $\gamma_A > \gamma_B$ ◯ $\gamma_A = \gamma_B$ ◯ $\gamma_A < \gamma_B$ ◯ Cannot draw any conclusion.

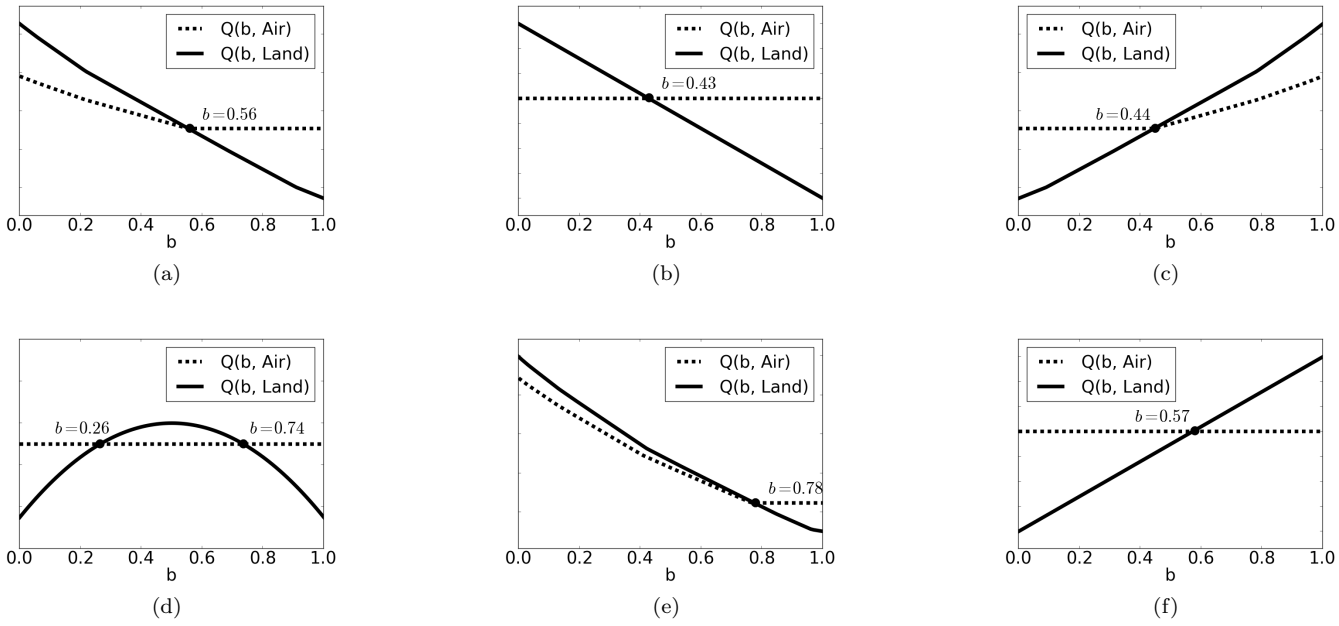


Figure 3: Q-values as a function of belief state.

(ii) [3 pts] The plots in Figure 3 claim to show the optimal Q-values for each action as a function of the belief state b (i.e. the probability that bandits are present), under an infinite time horizon with some discount factor γ. In reality, three of the graphs contain the true Q-values for $\gamma = 0$, $\gamma = 0.5$, and $\gamma = 0.9$, while the other three graphs are totally made up and don't correspond to Q-values under any value of γ. Each intersection between the two Q-functions is labeled with the belief state at that point. Note that the plots are intended to show qualitative behavior, so different plots do not necessarily have the same scale on the y-axis.

For each value of the discount γ, fill in a single circle indicating which plot contains the Q-values under that discount. *This question can be answered without any computation.*

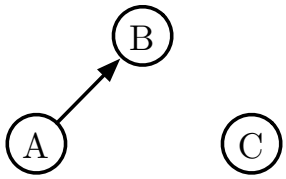
$\gamma = 0$:	◯ (a)	◯ (b)	◯ (c)	◯ (d)	◯ (e)	◯ (f)
$\gamma = .5$:	◯ (a)	◯ (b)	◯ (c)	◯ (d)	◯ (e)	◯ (f)
$\gamma = .9$:	◯ (a)	◯ (b)	◯ (c)	◯ (d)	◯ (e)	◯ (f)

Q8. [12 pts] Bayes' Nets Representation

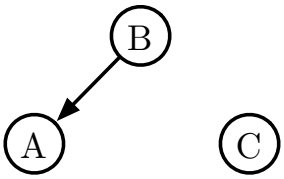
(a) The figures below show pairs of Bayes' Nets. In each, the **original** network is shown on the left. The **reversed** network, shown on the right, has all the arrows reversed. Therefore, the reversed network may not be able to represent all of the distributions that the original network is able to represent.

For each pair, add the *minimal* number of arrows to the **reversed** network such that it is guaranteed to be able to represent all the distributions that the **original** network is able to represent. If no arrows need to be added, clearly write "none needed."

(i) [2 pts]

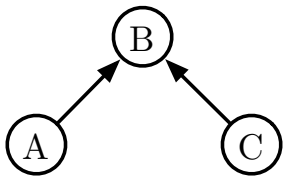


Original

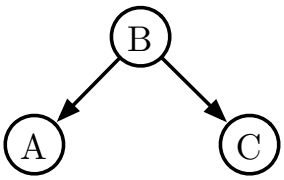


Reversed

(ii) [2 pts]

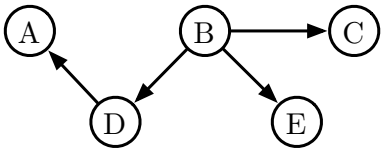


Original

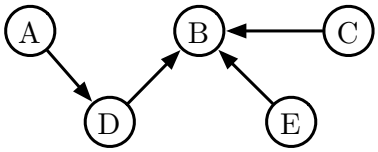


Reversed

(iii) [2 pts]



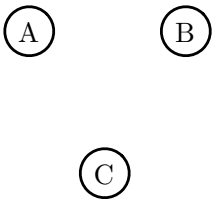
Original



Reversed

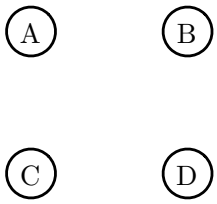
(b) For each of the following Bayes' Nets, add the *minimal* number of arrows such that the resulting structure is able to represent all distributions that satisfy the stated independence and non-independence constraints (note these are constraints on the Bayes net *structure*, so each non-independence constraint should be interpreted as disallowing all structures that make the given independence assumption). If no arrows are needed, write "none needed." If no such Bayes' Net is possible, write "not possible."

Use a pencil or think twice! Make sure it is very clear what your final answer is.



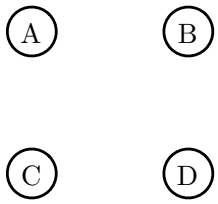
(i) [2 pts] **Constraints:**

- $A \perp\!\!\!\perp B$
- not $A \perp\!\!\!\perp B \mid \{C\}$



(ii) [2 pts] **Constraints:**

- $A \perp\!\!\!\perp D \mid \{C\}$
- $A \perp\!\!\!\perp B$
- $B \perp\!\!\!\perp C$
- not $A \perp\!\!\!\perp B \mid \{D\}$

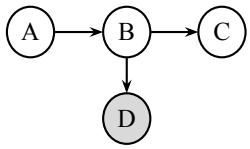


(iii) [2 pts] **Constraints:**

- $A \perp\!\!\!\perp B$
- $C \perp\!\!\!\perp D \mid \{A, B\}$
- not $C \perp\!\!\!\perp D \mid \{A\}$
- not $C \perp\!\!\!\perp D \mid \{B\}$

Q9. [11 pts] Preprocessing Bayes' Net Graphs for Inference

For (a) and (b), consider the Bayes' net shown on the right. You are given the structure but you are not given the conditional probability tables (CPTs). We will consider the query $P(B|+d)$, and reason about which steps in the variable elimination process we might be able to execute based on just knowing the graph structure and not the CPTs.



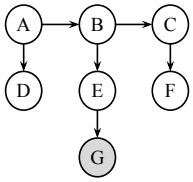
(a) [1 pt] Assume the first variable we want to eliminate is A and the resulting factor would be $f_1(B)$. Mark which one of the following is true.

- ◯ $f_1(+b) = f_1(-b) = 1$
- ◯ $f_1(+b) = f_1(-b) = 0$
- ◯ $f_1(B)$ *cannot* be computed from knowing only the graph structure.
- ◯ $f_1(B)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

(b) [1 pt] Assume the first variable we eliminate is C and the resulting factor is $g_1(B)$. Mark which one of the following is true.

- ◯ $g_1(+b) = g_1(-b) = 1$
- ◯ $g_1(+b) = g_1(-b) = 0$
- ◯ $g_1(B)$ *cannot* be computed from knowing only the graph structure.
- ◯ $g_1(B)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

For (c) through (g), consider the Bayes' net shown on the right. You are given the structure but you are not given the conditional probability tables (CPTs). We will consider the query $P(B|+g)$, and again we will reason about which steps in the variable elimination process we might be able to execute based on just knowing the graph structure and not the CPTs.



(c) [1 pt] Assume the first variable we eliminate is D and the resulting factor is $h_1(A)$. Mark which one of the following is true.

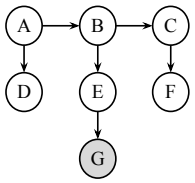
- ◯ $h_1(+a) = h_1(-a) = 1$
- ◯ $h_1(+a) = h_1(-a) = 0$
- ◯ $h_1(A)$ *cannot* be computed from knowing only the graph structure.
- ◯ $h_1(A)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

(d) [1 pt] Assume the first 2 variables we eliminate are A and D and the resulting factor is $i_2(B)$. Mark which one of the following is true.

- ◯ $i_2(+b) = i_2(-b) = 1$
- ◯ $i_2(+b) = i_2(-b) = 0$
- ◯ $i_2(B)$ *cannot* be computed from knowing only the graph structure.
- ◯ $i_2(B)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

(e) [1 pt] Assume the first variable we eliminate is F and the resulting factor is $j_1(C)$. Mark which one of the following is true.

- ◯ $j_1(+c) = j_1(-c) = 1$
- ◯ $j_1(+c) = j_1(-c) = 0$
- ◯ $j_1(C)$ *cannot* be computed from knowing only the graph structure.
- ◯ $j_1(C)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.



For your convenience we included the Bayes' Net structure again on this page.

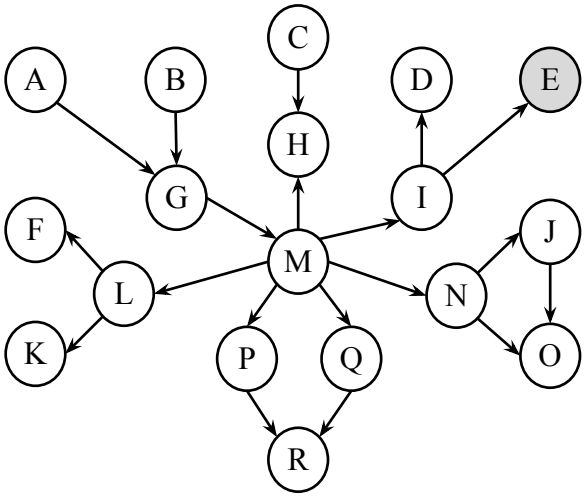
(f) [1 pt] Assume the first 2 variables we eliminate are $F$and C and the resulting factor is $k_2(B)$. Mark which one of the following is true.

- ◯ $k_2(+b) = k_2(-b) = 1$
- ◯ $k_2(+b) = k_2(-b) = 0$
- ◯ $k_2(B)$ *cannot* be computed from knowing only the graph structure.
- ◯ $k_2(B)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

(g) [1 pt] Assume the first variable we eliminate is E and the resulting factor is $l_1(B, +g)$. Mark which one of the following is true.

- ◯ $l_1(+b, +g) = l_1(-b, +g) = 1$
- ◯ $l_1(+b, +g) = l_1(-b, +g) = 0$
- ◯ $l_1(B, +g)$ *cannot* be computed from knowing only the graph structure.
- ◯ $l_1(B, +g)$ *can* be computed from knowing only the graph structure but is not equal to any of the provided options.

(h) [4 pts] In the smaller examples in (a) through (g) you will have observed that sometimes a variable can be eliminated without knowing any of the CPTs. This means that variable's CPT does not affect the answer to the probabilistic inference query and that variable can be removed from the graph for the purposes of that probabilistic inference query. Now consider the following, larger Bayes' net with the query $P(Q| + e)$.



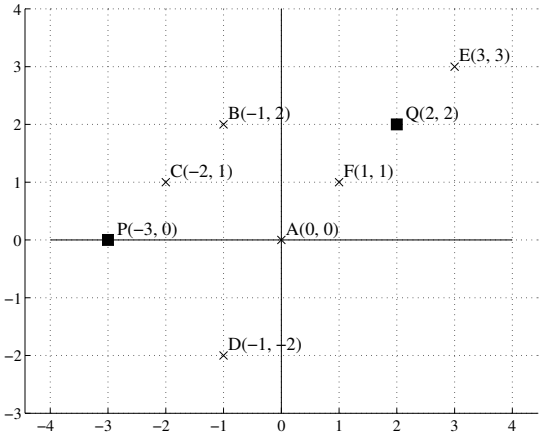
Mark all of the variables whose CPTs do *not* affect the value of $P(Q| + e)$:

◯ A	◯ B	◯ C	◯ D	◯ E
◯ F	◯ G	◯ H	◯ I	◯ J
◯ K	◯ L	◯ M	◯ N	◯ O
◯ P	◯ Q	◯ R		

Q10. [8 pts] Clustering

In this question, we will do k-means clustering to cluster the points $A, B \ldots F$ (indicated by $\times$'s in the figure on the right) into 2 clusters. The current cluster centers are P and Q (indicated by the ■ in the diagram on the right). Recall that k-means requires a distance function. Given 2 points, $A = (A_1, A_2)$ and $B = (B_1, B_2)$, we use the following distance function $d(A, B)$ that you saw from class,

$$d(A, B) = (A_1 - B_1)^2 + (A_2 - B_2)^2$$



(a) [2 pts] **Update assignment step:** Select all points that get assigned to the cluster with center at P:

◯ A ◯ B ◯ C ◯ D ◯ E ◯ F ◯ No point gets assigned to cluster P

(b) [2 pts] **Update cluster center step:** What does cluster center P get updated to?

Changing the distance function: While k-means used Euclidean distance in class, we can extend it to other distance functions, where the assignment and update phases still iteratively minimize the total (non-Euclidian) distance. Here, consider the Manhattan distance:

$$d'(A, B) = |A_1 - B_1| + |A_2 - B_2|$$

We again start from the original locations for P and Q as shown in the figure, and do the update assignment step and the update cluster center step using Manhattan distance as the distance function:

(c) [2 pts] **Update assignment step:** Select all points that get assigned to the cluster with center at P, under this new distance function $d'(A, B)$.

◯ A ◯ B ◯ C ◯ D ◯ E ◯ F ◯ No point gets assigned to cluster P

(d) [2 pts] **Update cluster center step:** What does cluster center P get updated to, under this new distance function $d'(A, B)$?

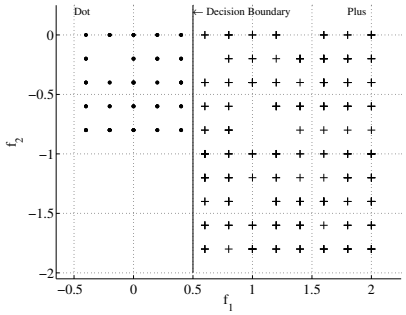
Q11. [6 pts] Decision Trees

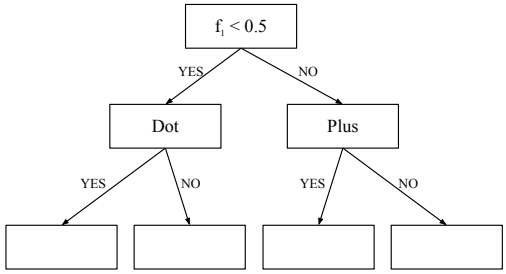
You are given points from 2 classes, shown as +'s and ·'s. For each of the following sets of points,

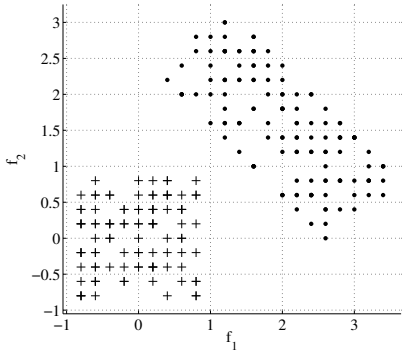
1. Draw the decision tree of depth at most 2 that can separate the given data completely, by filling in binary predicates (which only involve thresholding of a *single* variable) in the boxes for the decision trees below. If the data is already separated when you hit a box, simply write the class, and leave the sub-tree hanging from that box empty.

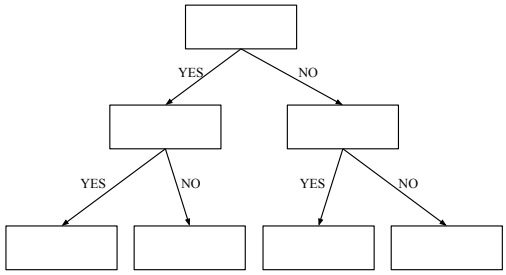
2. Draw the corresponding decision boundaries on the scatter plot, and write the class labels for each of the resulting bins somewhere inside the resulting bins.

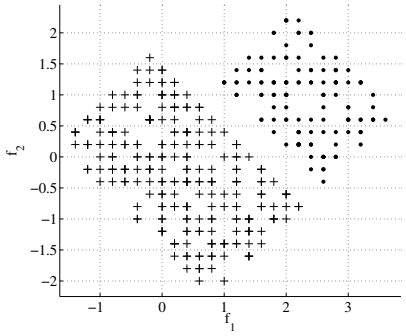
If the data can not be separated completely by a depth 2 decision tree, simply cross out the tree template. We solve the first part as an example.

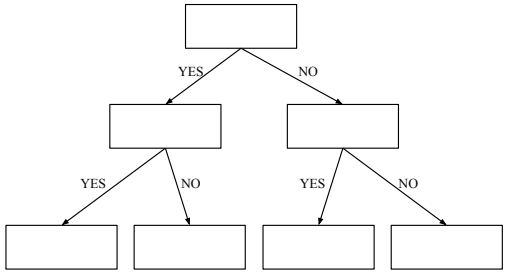


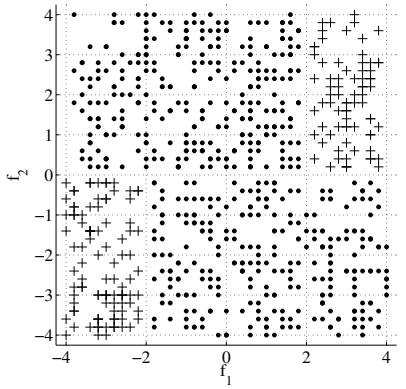


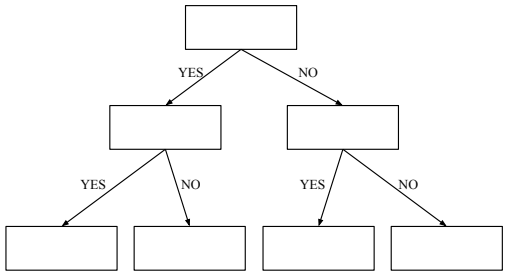








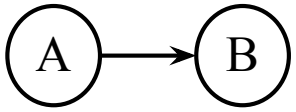




Q12. [10 pts] Machine Learning: Overfitting

Suppose we are doing parameter estimation for a Bayes' net, and have a training set and a test set. We learn the model parameters (probabilities) from the training set and consider how the model applies to the test set. In this question, we will look at how the average log likelihood of the training dataset and the test dataset varies as a function of the number of samples in the training dataset.

Consider the Bayes' net shown on the right. Let $x_1 \ldots x_m$ be the m training samples, and $x'_1 \ldots x'_n$ be the n test samples, where $x_i = (a_i, b_i)$. Recall that once we have estimated the required model parameters (conditional probability tables) from the training data, the likelihood L for any point x_i is given by:



$$L(x_i) = P(x_i) = P(a_i)P(b_i|a_i)$$

We additionally define the *log-likelihood* of the point x_i, to be $LL(x_i) = \log(L(x_i))$.

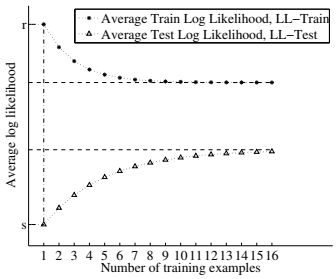
We use the log likelihood for a single point to define the average log likelihood on the training set (LL-Train) and the test set (LL-Test) as follows:

$$\text{LL-Train} = \frac{1}{m}\sum_{i=1}^{m} LL(x_i) = \frac{1}{m}\sum_{i=1}^{m} \log\left(L(x_i)\right) = \frac{1}{m}\sum_{i=1}^{m} \log\left(P(a_i)P(b_i|a_i)\right)$$

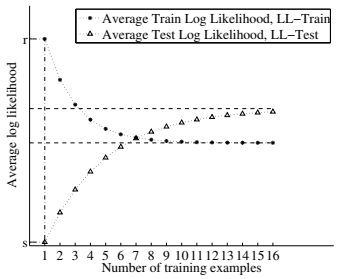
$$\text{LL-Test} = \frac{1}{n}\sum_{i=1}^{n} LL(x'_i) = \frac{1}{n}\sum_{i=1}^{n} \log\left(L(x'_i)\right) = \frac{1}{n}\sum_{i=1}^{n} \log\left(P(a'_i)P(b'_i|a'_i)\right)$$

We assume the test set is very large and fixed, and we will study what happens as the training set grows.

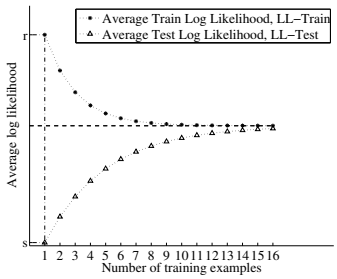
Consider the following graphs depicting the average log likelihood on the Y-axis and the number of training examples on the X-axis.



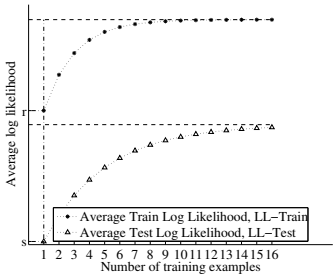
(a) A



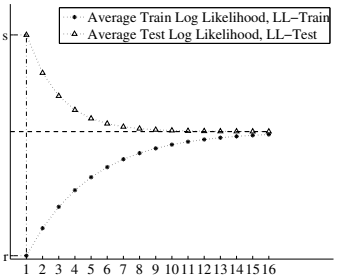
(b) B



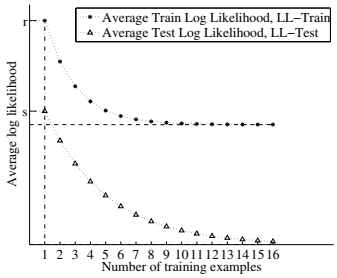
(c) C



(d) D



(e) E



(f) F

(a) [2 pts] Which graph most accurately represents the typical behaviour of the average log likelihood of the training and the testing data as a function of the number of training examples?

○ *A* ○ *B* ○ *C* ○ *D* ○ *E* ○ *F*

(b) [2 pts] Suppose our dataset contains exactly one training example. What is the value of LL-Train?

◯ $-\infty$ ◯ -2 ◯ -1 ◯ $-1/2$ ◯ 0

◯ $1/2$ ◯ 1 ◯ 2 ◯ ∞

◯ *Can* be determined but is not equal to any of the provided options

◯ *Cannot* be determined from the information provided

(c) [3 pts] If we did Laplace Smoothing with $k = 5$, how would the values of LL-Train and LL-Test change in the limit of infinite training data? Mark all that are correct.

◯ LL-Train would remain the same.

◯ LL-Train would go up.

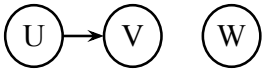
◯ LL-Train would go down.

◯ LL-Test would remain the same.

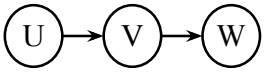
◯ LL-Test would go up.

◯ LL-Test would go down.

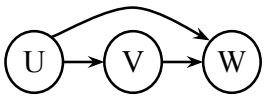
(d) [3 pts] Consider the following increasingly complex Bayes' nets: G_1, G_2, and G_3.



G_1

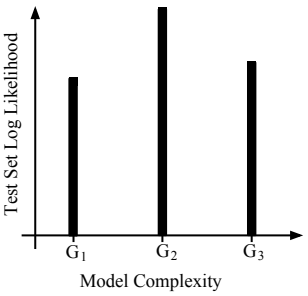


G_2

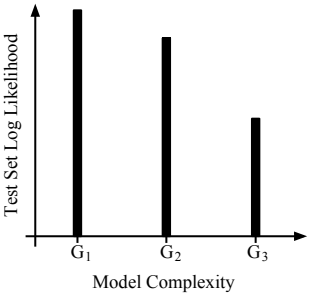


G_3

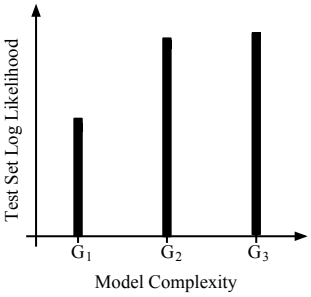
Consider the following graphs which plot the test likelihood on the Y-axis for each of the Bayes' nets G_1, G_2, and G_3



A



B



C

For each scenario in the column on the left, select the graph that best matches the scenario. Pick each graph exactly once.

Small amount of training data: ◯ *A* ◯ *B* ◯ *C*

Medium amount of training data: ◯ *A* ◯ *B* ◯ *C*

Large amount of training data: ◯ *A* ◯ *B* ◯ *C*