Question 1 – True/False – 30 points

Circle the correct answer each True / False question.

1. [✓ True] / False – A* Graph Search requires a consistent heuristic for optimality. (3 pt)

2. [✓ True] / False – In blind search (uninformed search), each node in a search tree corresponds to a node in the state graph. (3 pt)

3. True / [✓ False] – Alpha-Beta pruning is an exact algorithm that has been observed to, in practice, double the depth of the minimax tree that can be built. (3 pt)

4. [✓ True] / False – Policy Iteration always find the optimal policy, when run to convergence. (3 pt)

5. [✓ True] / False – Q-learning with constant, ϵ-greedy exploration ($\epsilon = 0.05$) will always converge to the optimal policy. (3 pt)

   must gradually decrease this, but not too quickly

   have to explore each state, action pair infinitely many times.

6. True / [✓ False] – Adding more edges to a Bayesian network can restrict the space of possible distributions it can represent. (3 pt)

7. True / [✓ False] – For estimating conditional queries in Bayesian networks, rejection sampling has generally been observed to provide better estimates that likelihood weighting (for a fixed number of samples). (3 pt)

8. [✓ True] / False – Naive Bayes models always encode incorrect independence assumptions. (3 pt)

9. True / [✓ False] – The order of elimination does not matter in variable elimination in Bayesian networks. (3 pt)

10. True / [✓ False] – Non-optimal paths are not useful in Q-learning. (3 pt)

    No transition model, so just keep trying...

Question 2 – Short Answer – 35 points

If using A* as TREE SEARCH, we use admissible heurestic. If using A* as GRAPH SEARCH, use Consistent heuristic

These short answer questions can be answered with a few sentences each.

1. Short Answer – Briefly describe the relationship between admissible and consistent heuristics. When would you use each, and why? (5 pts)

ADMISSIBLE -> $h(n) < h^*(n)$ where $h(n)$ is estimated cost and $h^*(n)$ is true cost. Never overestimates the cost of reaching goal.

CONSISTENT -> A consistent heurestic function estimates the distance of a given node to the goal node to be always at most equal to the estimated distance from any neighbouring node plus the cost of reaching that neighbor.

A CONSISTENT HEURESTIC IS ADMISSIBLE, but not the other way around

2. Short Answer – Briefly describe how you would decide which algorithm to use for answering queries to a Bayesian network. What is the key property of the network that, if known, would best help you make the appropriate decision. (5 pts)

-> If Bayesian network is small and not complex (less interconnecting edges), we can use variable elimination. This will give us accurate answers. (worst case time complexity is too high)

-> Use sampling (Gibbs/likelihood weighting) when a complex bayesian model exists and you only want to account for Dependent nodes. This is an approximation algorithm, unlike inference by enumeration/variable elimination.

3. Short Answer – For Q-learning, when would you prefer to use linear function approximation and when you just use the tabular version? Is there ever any drawback to using the linear version? (5 pts)

in certain direction of a weight

If the state space is ridiculously large, it makes sense to encode similar states into certain features (Preferably normalized) with weights. For Q learning, we can write the q function as a linear combination of these features and weights. This allows us to nudge/adjust the weights of active features, so if something unexpected happens, we can disprefer all states with that states features. We would use a tabular version, if no correlation between states, and state space is small. The Drawback is that the states can share some features, but have very different values

4. Short Answer – Briefly describe the difference between UCS and A* search. When would you prefer to use each, and why? (5 pts)

UCS -> This search expands in all directions, and has no information about which direction the goal is towards. Complete & optimal but too much exploration. This is A* with $h(n) = 0$. Use when consistent heurestics are tough to derive, and optimality is important

Astar -> Search by expanding nodes that lead you towards the goal. Heurestic includes cost to nearest node ($g(n)$) and the cost from that node to the goal node. Use when heurestics are easy to design (Relaxed Problem)

5. Short Answer – In machine learning, explain generalization and over-fitting. Describe an experimental setup that correctly measures generalization. Assume that your algorithm has one hyperparameter that must be set. (5 pts)

In ML, we want to build a classifier which does well on Test data, and not just our training data. In other words, we want to generalize our model, and not overfit it to our training data. This can be done by smoothing or regularizing our estimates for Parameters. To measure generality, we split data into training, validation, and testing set. We learn Model Probabilities on training set, tune the hyperparameter on validation test, and compute accuracy on "test set".

6. Short Answer – Explain the Bayes rule and mention two methods (discussed in the class) that use Bayes rule. (5 pts)

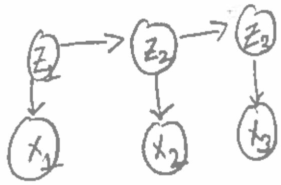
Bayes Rule allows us to update our beliefs abat hylothesis A in light of Evidence B. Specifically, our Posterior belief P(A|B) is calculated by multiplying Prior belief P(A) with likelihood P(B|A) that B occurs given A is true

$\longrightarrow P(A|B) = P(B|A)P(A) / P(B)$

Naivee Bayes, Variable Elimination and Hidden Markov Models use Bayes' Rule

7. Short Answer – Describe the forward algorithm in HMMs and explain why it is useful. (5 pts)

Forward algorithm is used to do inference in Hidden Markov Models. It is basically a case of Dynamic Programming. In the below HMM..



if given $P(X_k|Z_k)$, $P(Z_k|Z_{k-1})$, $P(Z_1)$

$\uparrow$ emmission Dist $\uparrow$ transition Dist. $\uparrow$ initial Dist

Forward Algorithm: Allows us to compute $P(Z_k | X_{1:k})$ $\forall k = 1 \ldots n$

using Recursion and conditional independance properties.

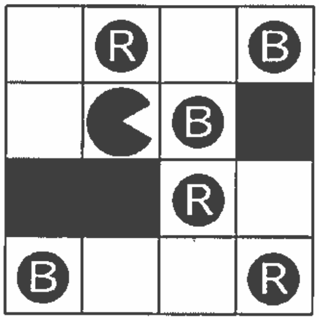
$$\underline{P(Z_k, X_{1..k})} = \sum_{Z_{k-1}} P(X_k|Z_k) P(Z_k|Z_{k-1}) \underline{P(Z_{k-1}, X_{1..k-1})}$$

Recurse

It is useful because its time comlexity is $\theta(nm^2)$ because $\theta(m)$ for each Z_k, there are m Z_k's .. so $\theta(m^2)$ for each k. We have to do this n times .. Compare this to naivee approach where it would be $\theta(m^n)$.

Question 3 – Ordered Pacman Search – 25 points

Consider a new Pacman game where there are two kinds of food pellets, each with a different color (red and blue). Pacman has peculiar eating habits; he strongly prefers to eat all of the red dots before eating any of the blue ones. If Pacman eats a blue pellet while a red one remains, he will incur a cost of 100. Otherwise, as before, there is a cost of 1 for each step and the goal is to eat all the dots. There are K red pellets and K blue pellets, and the dimensions of the board are N by M.



$K = 3$, $N = 4$, $M = 4$

Model specific.
to this question

1. Give a tight upper bound on the size of the state space required to model this problem. Briefly describe your reasoning. [10 pts]

13×2^6 The actual State doesn't need to encode which direction the Pacman is facing.

$(N \times M - 3) \times 2^{2K}$

2. Give a tight upper bound on the branching factor of the state space. Briefly describe your reasoning. [5 pts]

3 ← it can go {N, S, E, N}, but in the grid above there is no where it has 4 options.

3. Which search algorithm would pacman execute to get the optimal path? Why? (describe in one or two sentences) [5 pts]

You can use either UCS OR A-Star. Use UCS if you don't have a consistent heurestic, and just want something complete and optimal at the cost of high computational complexity. USE A* with consistent heurestic if you want less computational complexity and an optimal solution

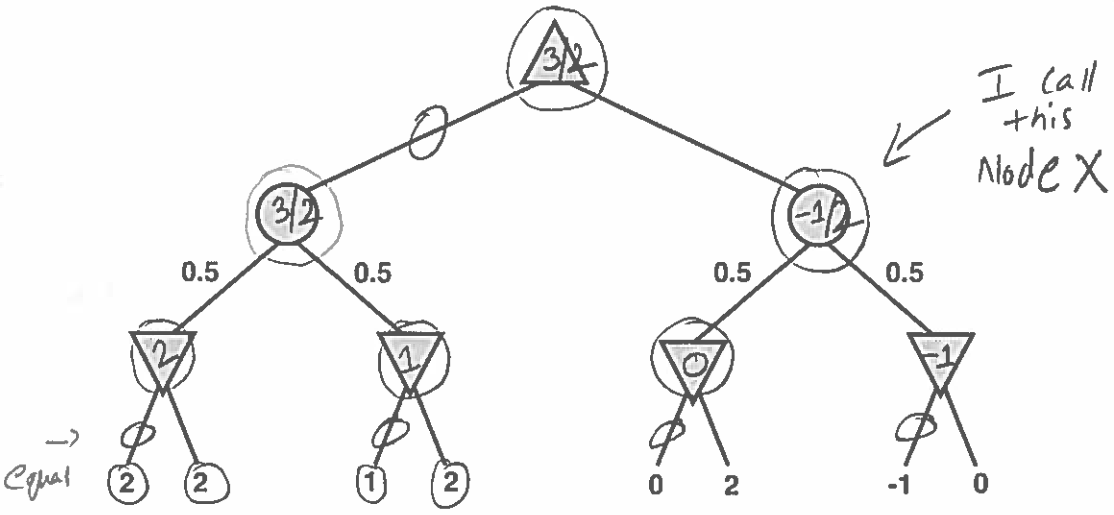
4. Give an admissible heuristic for this problem. [5 pts]

① Run BFS to find a Path which eats all the Remaing nodes (optimally)
The length of that Path will be always less than OR equal to the optimal Path

② Largest Manhattan distance 6 from all Remaing food Pellets

Question 4 – Game Trees – 30 points

Consider the following game tree, which has min (down triangle), max (up triangle), and expectation (circle) nodes:



1. In the figure above, label each tree node with its value (a real number). [7 pts]
2. In the figure above, circle the edge associated with the optimal action at each choice point. [7 pts]
3. If we knew the values of the first six leaves (from left), would we need to evaluate the seventh and eighth leaves? Why or why not? [5 pts]

   We'll need to evaluate the 7th leaf and since its less then the the best available value to the maximizer, we can prune the 8th leaf (i.e $\frac{3}{2} < -1$)

4. Suppose the values of leaf nodes are known to be in the range [−2, 2], inclusive. Assume that we evaluate the nodes from left to right in a depth first manner. Can we now avoid expanding the whole tree? If so, why? Circle all of the nodes that would need to be evaluated (include them all if necessary). [11 pts]

We can stop at Node X as the max value the Right side of Node X could return is 2, and the left hand side already returned 0. Multiplying 1/2 by 2 would give you 1 which is less than best available option to Maximizer, i.e 1.5, so we dont evaluate any further

Question 5 – MDPs – 20 points

Consider the following elevator scenario, where you are a rider trying to leave a building at the end of the day.

The building has four floors (ranging in numbers from 1-4) and there is one elevator with four buttons, one for each floor. After a button is pressed, the elevator will move directly to the desired floor 80% of the time, but will move to one of the other two floors with equal probability. For example, if the elevator is on floor 3 and the rider presses 4, there is a 80% chance of arriving at floor 4, 10% chance of floor 2, and 10% chance of floor 1.

In general, the rider will have an equal probably of starting out at floors 2-4, but never starts on floor 1. Furthermore, this is a toll elevator. It costs 10 cents every time you press a button. Finally, since it is late in the day, we will assume that the rider wants to get to floor 1 to go home (stop riding).

1. Model this problem as an MDP. Specify all of the necessary parameters. [15 pts]

States -> {F1, F2, F3, F4} where F is FlooR

TeRMINAL State -> {F1}

S_0 -> {F2, F3, F4}
(Starting state)

Actions -> {B1, B2, B3, B4, Done} where B_i means PRess Button i and "Done" only applies to state 1

Rewards -> $R(F_i, B_j, F_k) = -0.10$ (PRessing any buttOn when not in state 1)
$i, j, k \in [1,2,3,4]$ and $i \neq 1$

-> $R(F1) = 0$

$\gamma = 1$ (no Discounting)

$T(F_j, B_i, F_i) = 0.80$ where $j \neq 1$

$T(F_j, B_i, F_k) = 0.10$ where $i \neq k$ & $j \neq 1$

In state FlooR1 OR F1, theRe is ONLY 1 action, DONE

2. What is the optimal policy for this problem? [5 pts]

We can Run value iteration . . . , but its PRetty obvious the optimal Policy is to PRess 1 regaRdless of which floor you are in . . Using value iteration, we would get . .

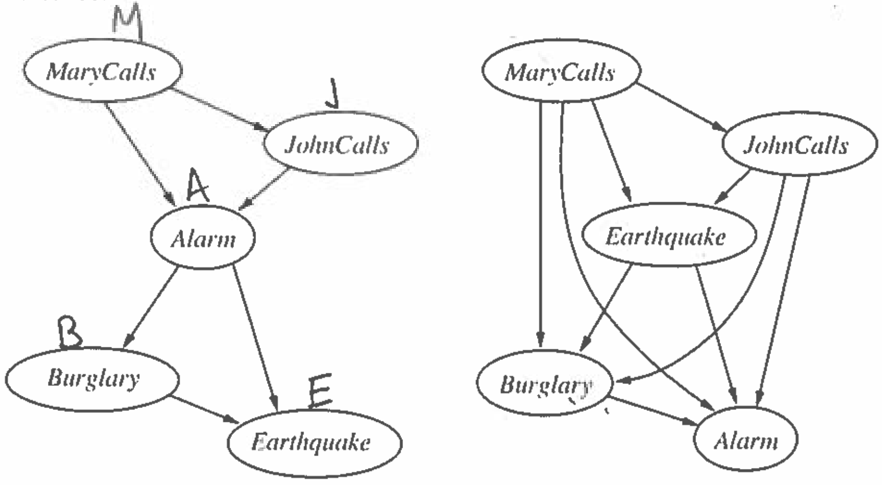
$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

$V_0(F_{2|3|4}) = (.80)(-.10+0) + .10(-.10+0) + .10(-.10+0) = \boxed{-0.1}$

$V_0(F_1) = \boxed{0}$ <- go to F_1

Question 6 – Bayesian Networks – 30 points

Consider the following two Bayesian networks, which are variations on the alarm network we discussed in class:



(a) (b)

1. Based on the network structure alone, which network above makes the most independence assumptions? [3 pts]

(A)

2. Write down two conditional independence assumptions encoded by the structure of network (a). If there are not two, write as many as possible. [6 pts]

(4) Johncalls $\perp\!\!\!\perp$ Burglary | Alarm

(6) MARYcalls $\perp\!\!\!\perp$ Alarm | John

(2) Johncalls $\perp\!\!\!\perp$ Earthquake | Alarm

(1) MARYCALLS $\perp\!\!\!\perp$ BuRgulary | Alarm (3) MARYcalls $\perp\!\!\!\perp$ Earthquake | Alarm

3. Write down two conditional independence assumptions encoded by the structure of network (b). If there are not two, write as many as possible. [6 pts]

can't find anything. There is an arrow from every node to every other node destroying all conditional indelendence assumptions.

4. Simulate the execution of the variable elimination algorithm on network (a) to compute $P(Marycalls|Burglary = true)$. Since we have not given you the CPTs, you do not need to compute the entries. Instead, just list the tables that would be created and eliminated at each step of the computation. Use the most computationally efficient variable ordering. [15 pts]

→

#4) Init factures : $\underline{P(M)}\ \underline{P(J|M)}\ \underline{P(A|M,J)}\ \underline{P(B=true|A)}\ \underline{P(E|B=true, A)}$

ORDERing: E, J, A

a) Eliminate E

$\rightarrow f_1(B=true, A) = \sum_e P(e|B=true, A)$

Table created ->

B	A	f_1
+B	+A	...
+B	-A	....

Table eliminated -> $P(\cancel{E|B=true}|A)$

b) Eliminate j

$\rightarrow f_2(M,A) = \sum_j P(j|M)\, P(A|M,j)$

Table created ->

M	A	f_2
+M	+A	....
+M	-A	....
-M	+A	...
-M	-A	...

Table(s) Eliminated -> $P(\cancel{J|M})$, $P(\cancel{A|M},j)$

c) Eliminate A

$\rightarrow f_3(B=true, M) = \sum_a f_2(M,a)\, f_1(B=true, a)\, P(B=true|a)$

Table created ->

B	M	f_3
+B	+M	..
+B	-M	.

Table(s) Eliminated -> $P(\cancel{B=true}|A)$, $f_2(\cancel{M,A})$, $f_1(\cancel{B=true},A)$

Left with $f_3(B=true, M)$ & $P(M)$:

$$\boxed{P(M, B=true) = f_3(B=true, M)\, P(M)}$$

Renormalize ↑ to get $P(M|B=true)$