

**ID:** 00110001 00110000 00110000

1. (12 points) True or False :

True In the particle filtering version we have seen for inference in HMMs, effective number of particles may change over time.

False Training data performance is always a good indicator of the general performance of an ML model.

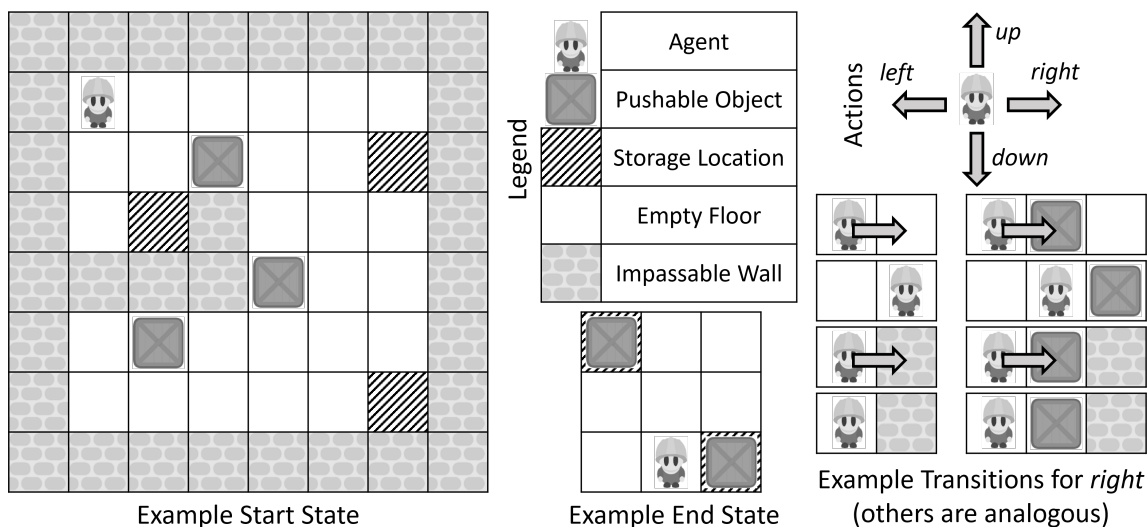
False One way to overcome underfitting issues in Machine Learning is to get more data.

True Expectiminimax algorithm can be modified to solve an MDP.

True Policy iteration is a better choice than value iteration if the number of possible actions are large.

False The transition model must be known to be able to carry out Q-learning.

2. (10 points) Sokoban is a puzzle game in which an agent pushes around objects in a maze, trying to get these pushable objects to storage positions. The game is played on a grid. Each grid location can either be a wall or a floor. Some floor locations contain objects and some storage locations. The agent can be on a floor unless it contains an object. The agent can move in 4 cardinal directions on the grid (*up*, *down*, *left*, *right*). If there is an object on a grid location that the agent is trying to move to, then it pushes the object as long as the next grid location is not a wall. If there is a wall, then nothing happens. The agent cannot pull the objects. The game ends when all the objects are on the storage locations. See the below figure for a visual description:



Formulate Sokoban as a search problem. Which search algorithm would you chose to use and why? What would be your cost and/or heuristics if your algorithm requires them?

(Please provide your solution on the following page)

(Page reserved for the solution of question 2)

One way to formulate it is given below.

**State Space** ( $S$ ): Each element in the state space is a tuple  $(F, O, L, AG)$  where (2 points):

- $F$  is a boolean matrix representing the grid. Each entry represents wall or floor
- $O$  is a list of object locations. Each location is 2D.
- $L$  is a list of target locations. Each location is 2D.
- $AG$  is the agent location in 2D.

**Actions** ( $A$ ): As given above up, down, left, right

**Transition Model** ( $T : S \times A \rightarrow S$ ) (2 points along with the actions): Most of it is written in the text. The main point is to point out that this is part of the formulation and it is about the states and actions. The actions only change the subset of the state space ( $AG$  and  $O$ ).

**Cost** ( $S \times A \rightarrow \mathbb{R}$ ): Each action can be taken as unit cost. (0.5 points)

**Start State** ( $s_0$ ): The given start space (0.5 points)

**Goal Test** ( $g(s)$ ): Returns true if the objects ( $O$ ) are on the targets ( $L$ ), else returns false (1 point)

**Bonus**: Add an extra function to detect when the state would never lead to a solution (e.g. when an object is pushed to a corner) (1 point). This can be used to prune the state space to be searched.

**Algorithm**: The only choice is  $A^*$  (based on what we have seen in the class). Sokoban definitely requires a good heuristic to be solved due to its state space complexity. You should be able to come to this conclusion. Note that even with  $A^*$ , Sokoban is not easy to solve for complex and/or medium sized mazes (1 point for the algorithm, 1 point for reasoning).

**Heuristics**: Cost of moving the agent to all un-occupied target locations. Variations include ignoring objects and/or ignoring walls. Additional heuristics would be the distance of each object to its closest target location. Other heuristics can be built by combining ideas (move agent to objects, then objects to locations) (2 points for a good admissible heuristic, 1 point for anything that makes sense)

3. (8 points) You are given six variables,  $A, B, C, D, E, F$  each with the domain  $\{1, 2, 3, 4, 5, 6\}$ . You are also given the the following constraints on the variables:

1. All different values
2.  $A > 3$
3.  $B < 5$
4.  $E$  is odd
5.  $E + F < 5$
6.  $C + D$  is even
7.  $A + B$  is divisible by 3 ( $(A + B)/3$  is an integer)

You are asked to use **hill-climbing** (or in this case hill-descending) local search. The objective function to minimize is the number of violated constraints. The allowable action at each step is to perform a single value swap between two variables (e.g. swapping  $A = 1$  and  $B = 2$  would result in  $A = 2$  and  $B = 1$ ).

You are going to fill the table below, given the initial assignment. Note that with the initial assignment and the action definition, you do not need to worry about constraint 1. At each step, write the current assignment, the value of the objective function and circle the two variables that will be swapped. The size of the table is not an indication of the solution length. You can extend it if you need to or finish before filling it up. Break all ties in alphabetically by looking at all variables between swaps (e.g. if swapping  $A - D$  and  $B - C$  make the same improvement, pick  $A - D$  since  $A$  comes before all the other letters). Stop when you hit a local minima and indicate it as such.

Var	Init.	Step 1	Step 2	Step 3	Step 4	Step 5
A	4	4	5			
B	5	5	4			
C	1	2	2			
D	6	6	6			
E	2	1	1			
F	3	3	3			
Obj	4	1	0			

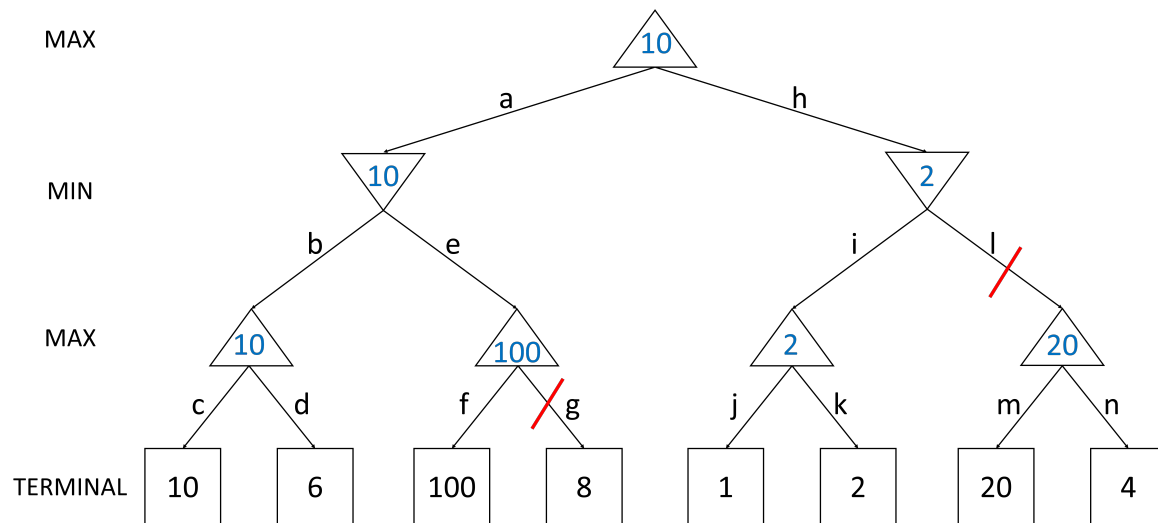
- Init: Cost 4, violating c3,c4,c5,c6, swap  $C - E$  (quick heuristic: try swapping out  $E$  first since it is in most violations)
- Step 1 Cost 1, violating c3, swap  $A - B$
- Step 2 Cost 0, violation

Cost calculations: 3 (1 per)

Correct swaps: 5 (2.5 per)

Wrong swap that still decreases the cost: 1 point, up to 3 swaps. Cost calculation will be accepted as if the swap was correct.

4. (10 points) You are given the game tree below



- (a) (2 points) Fill in the values of each state on the tree. Mark the optimal first move of the player MAX.

See the figure. 7 values + 1 move, 0.25 points each

- (b) (8 points) Perform alpha-beta pruning, exploring moves from left to right. Mark/cross-out the pruned arcs (or circle the never-explored subtrees). You do not need to keep track of all the alpha and beta values but describe why you pruned a given arc.

See the figure for the pruned arcs (**g** and **l**).

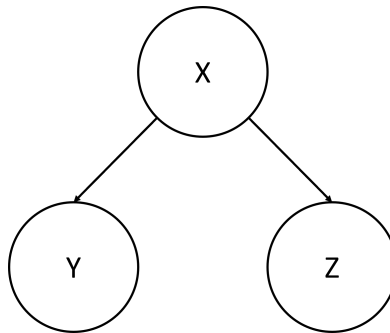
**g** is pruned since MIN would not chose **e** over **b** after seeing 100 ( $100 > 10$ ).

**l** is pruned since MAX would not chose **h** over **a** after seeing 2 ( $2 < 10$ ).

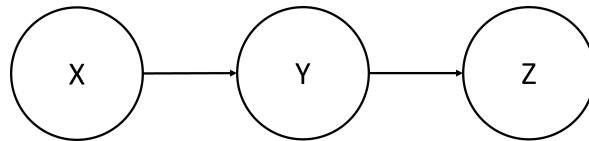
2 points per arc and 2 points per correct explanation

5. (10 points) You have observed multiple samples of three binary random variables  $X$ ,  $Y$  and  $Z$ . You want to build a Bayesian Network (BN) based on your measurements. Answer the questions below.

- (a) (2 points) Draw the BN with the assumption that the  $X$  is the cause of both  $Y$  and  $Z$ , and that  $Y$  and  $Z$  are independent given  $X$ .



- (b) (2 points) Draw the BN with the assumption that  $X$  is the only direct cause of  $Y$  and  $Y$  is the only direct cause of  $Z$ .



- (c) (4 points) How would you pick the BN structure between part (a) and part (b) given the observed samples? Explain your approach.

There are several approaches for this. I will present three of them. Two of the approaches require a closeness/distance “measure” between two discrete distributions to compare them but you are not going to be held responsible for the details.

**Assumption Checking:** Check the conditional independence assumptions against the empirical distributions. For example in part (a), this is  $P(Y, Z|X) = P(Y|X)P(Z|X)$ . We have the CPTs of the right hand side and can get the empirical distribution of the left for comparison.

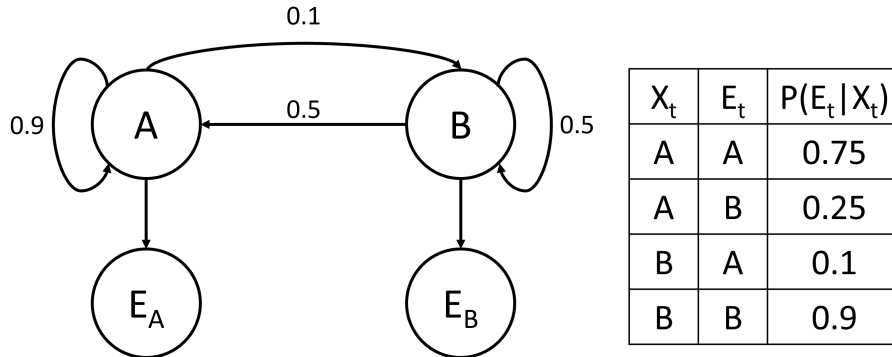
**Checking Against the Joint:** Use the samples to calculate the empirical joint distribution  $P(X, Y, Z)$ . Then we the joints obtained from both parts (multiply together the CPTs) for comparison. This step is not feasible if there are a lot of random variables.

**Checking the Data Likelihood:** Let  $D$  be the set of all collected samples. We calculate the data likelihood as  $\prod_{X,Y,Z \in D} P(X, Y, Z)$  for both the BNs and pick the larger one. Note that we are multiplying a lot of numbers between 0 and 1 which will lead to underflow. So it is better to compare the log likelihoods:  $\sum_{X,Y,Z \in D} \log(P(X, Y, Z))$

- (d) (2 points) Is there any advantage of using the collected samples versus using the learned BN for inference? Why or why not?

The simplest answer is that the data should be preferred since the assumptions made to build the BNs may be wrong. This effectively becomes “rejection sampling”. This answer will be accepted. On the other hand, if assumptions are correct (e.g. from domain knowledge), all the disadvantages of rejection sampling will apply here as well and as such the BNs should be preferred. This reasoning will also be accepted.

6. (15 points) Consider the two state ( $A$  or  $B$ ) Hidden Markov Model below. Let  $X$  denote the current state and  $t$  the current time step, then  $X_t = A$  or  $X_t = B$ . The prior distribution is uniform, i.e.,  $P(X_0 = A) = P(X_0 = B) = 0.5$ .



- (a) (5 points) A single time-step passes and you observe  $E_1 = A$ . What is the belief distribution  $B_1(X)$ ? (Writing the passage of time and observation steps separately would increase your chance of partial grade if you are not sure)

$$\begin{aligned}
 B_t(X = s) &= P(X_t = s | E_{1..t} = e_{1..t}) \propto P(E_t = e_t | X_t = s) P(X_t = s | E_{1..t-1} = e_{1..t-1}) \\
 &= P(E_t = e_t | X_t = s) B'_t(X = s) \\
 B'_t(X = s) &= P(X_t = s | E_{1..t-1} = e_{1..t-1}) = \sum_{s'} P(X_t = s | X_{t-1} = s') B_{t-1}(X = s') \\
 B_0(X = s) &= P(X_0 = s)
 \end{aligned}$$

$$\begin{aligned}
 B'_1(X = A) &= P(X_1 = A | X_0 = B) P(X_0 = B) + P(X_1 = A | X_0 = A) P(X_0 = A) \\
 &= 0.5 \cdot 0.5 + 0.9 \cdot 0.5 = 0.7 \\
 B'_1(X = B) &= P(X_1 = B | X_0 = B) P(X_0 = B) + P(X_1 = B | X_0 = A) P(X_0 = A) \\
 &= 0.5 \cdot 0.5 + 0.1 \cdot 0.5 = 0.3
 \end{aligned}$$

$$\begin{aligned}
 B_1(X = A) &\propto P(E_1 = A | X_1 = A) B'_1(X = A) = 3/4 \cdot 0.7 = 21/40 = 105/200 \\
 B_1(X = B) &\propto P(E_1 = A | X_1 = B) B'_1(X = B) = 0.1 \cdot 0.3 = 3/100 = 6/200
 \end{aligned}$$

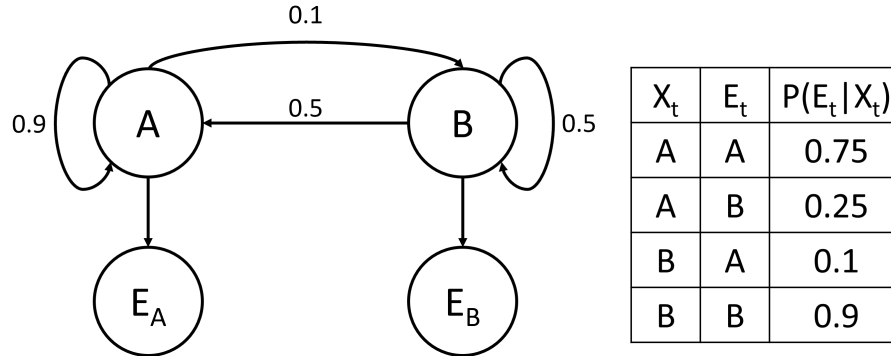
Single-step:

$$\begin{aligned}
 B_1(X = A) &\propto 0.75 \cdot (0.5 \cdot 0.5 + 0.9 \cdot 0.5) = 105/200 \\
 B_1(X = B) &\propto 0.1 \cdot (0.5 \cdot 0.5 + 0.1 \cdot 0.5) = 6/200
 \end{aligned}$$

Normalize to get:

$$B(X_t = A) = 105/111 \approx 0.946, B(X_t = B) = 6/111 \approx 0.054$$

- (b) (8 points) Let's go back to  $t = 0$ . This time we want to do particle filtering. We start with 4 particles as follows  $[A, B, A, B]$ . The model evolves for 1 step and we get  $E_1 = A$  as before. Apply one iteration of particle filtering. Go over the particles with the given order. Show all your work. Use the following uniform random variables whenever you need to for sampling: 0.42, 0.87, 0.53, 0.37, 0.28, 0.71, 0.96, 0.64. The figure is repeated for convenience



Current State	Time Lapsed State	Observed $w$	Normalized $w$	Resampled State
A	A	0.75	$15/47 \approx 0.32$	A
B	A	0.75	$15/47 \approx 0.32$	A
A	A	0.75	$15/47 \approx 0.32$	B
B	B	0.1	$2/47 \approx 0.04$	A

First step is to let the time pass, i.e., sample from the transition function (we will also accept the “other” version of the cumulative distributions as long as the solutions are consistent.)

- Sample from  $P(\cdot|X_{t-1} = A)$ . Since  $0 \leq 0.42 < 0.75$ , we get A (would still get A in the other version)
- Sample from  $P(\cdot|X_{t-1} = B)$ . Since  $0.5 \leq 0.87 < 1.0$ , we get A (would get B in the other version)
- Sample from  $P(\cdot|X_{t-1} = A)$ . Since  $0 \leq 0.53 < 0.75$ , we get A (would still get A in the other version)
- Sample from  $P(\cdot|X_{t-1} = B)$ . Since  $0 \leq 0.37 < 0.5$ , we get B (would get A in the other version)

We calculate the weights which is trivial ( $15/20 \times 3 + 2/20 = 47/20$ ). Then we normalize them which is also trivial.

Lastly, we need to re-sample. The way we build the cumulative distribution matters here as well. We will accept any consistent approach.

- First Particle:  $0 \leq 0.28 < 15/47$ , we get A
- Second Particle:  $30/47 \leq 0.71 < 45/47$ , we get A
- Third Particle:  $45/47 \leq 0.96 < 47/47$ , we get B (this one is very close but skews towards B. However, we will accept both)
- Fourth Particle:  $30/47 \leq 0.64 < 45/47$ , we get A (this one is also close but both sides are A)

An alternative is to sample A for between 0 and 45/47, which would get you the same samples. If you flip this, you would get all A's.

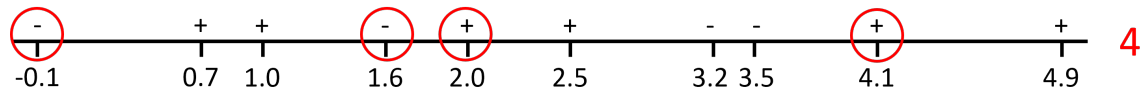
Time-lapse: 3, weighting and normalizing: 2, re-sampling: 3



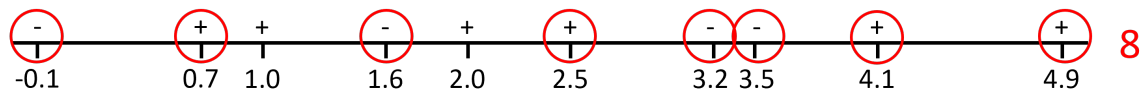
- (c) (2 points) What is the belief state distribution based on part (b)? This is just the ratio of particle type to the total number of particles. For our case:  $B_1(X = A) = 0.75$  and  $B_1(X = B) = 0.25$ .

7. (10 points) You are given a dataset (see the figures below) with a one dimensional real valued input and a binary output (+ or -). You are going to use k-Nearest Neighbors (k-NN) algorithm to predict the output given an input.

- (a) (2 points) What is the leave-one-out cross-validation (Pick 1 point as a test, remaining as training data, do it for all points) error of  $k = 1$  (looking at a single neighbor) on this dataset? Give your answer as the number of misclassifications. Circle the misclassifications in the figure below.



- (b) (3 points) What is the leave-one-out cross-validation error of  $k = 3$  (looking at 3 neighbors) on this dataset? Give your answer as the number of misclassifications. Circle the misclassifications in the figure below.



- (c) (4 points) Comment about the type of fit (overfit, underfit or goodfit) for both  $k$  values and state your reasons. Use your answers to previous parts and data noise if needed. For the k-NN method, we know that low  $k$  leads to overfitting and high  $k$  leads to underfitting.

For  $k = 1$ , we have no training misclassification (for the entire data). For  $k = 3$ , we have 6 training misclassifications (for the entire data).

Combining the cross-validation and the training performance, we can say that the  $k = 3$  leads to underfit for this problem.

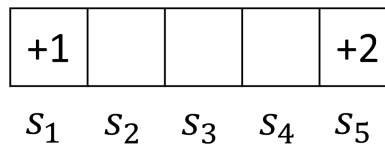
Combining the cross-validation and the training performance, the  $k = 1$  case looks to have overfitting.

Overall, it is difficult to argue with such a small dataset but the  $k = 3$  case is obvious.

- (d) (1 point) Based on your answers, which  $k$  would you pick?

$k = 1$  since it has a lower validation error.

8. (10 points) You are given the small grid-world along with a MDP definition below



- States:  $s_1, s_2, s_3, s_4, s_5$  ( $s_1$  and  $s_5$  are terminal states)
- Actions: Left( $L$ ), Right( $R$ )
- Transition Model: The actions move the agent in their direction with 0.9 probability and keep the agent still with 0.1 probability for non-terminal states For  $i = 2, 3, 4$ :
  - $P(s_{i-1}|s_i, L) = 0.9, P(s_i|s_i, L) = 0.1$
  - $P(s_{i+1}|s_i, R) = 0.9, P(s_i|s_i, R) = 0.1$
- Reward Function: -1 for transitioning to empty cells (including self-transition) and the given numbers for the other cells. The agent receives a reward when transitioning into the cells.
- Discount Factor:  $\gamma = 1.0$

You want to perform a single step of policy iteration.

- (a) (6 points) Policy Evaluation: Calculate the values of  $s_2, s_3, s_4$  for “always go Left” policy ( $\pi_L$ ). The values of the terminal states are 0. Hint: A closed form solution exists, start by writing the Bellman equation for  $s_2$ . Show your work for full points.

$$V^{\pi_L}(s_2) = \underline{8/9} \qquad V^{\pi_L}(s_3) = \underline{-2/9} \qquad V^{\pi_L}(s_4) = \underline{-12/9}$$

We want to calculate the value function for the given policy. As a result, we are not going to use the Bellman Optimality Equations but will use Bellman Value Equations, as described in the policy evaluation part of the course. Also note that the reward function has the form  $R(s')$  since it only depends on the state we transition to.

$$V^{\pi_L}(s) = \sum_{s'} P(s'|s, L)(R(s') + \gamma V^{\pi_L}(s')).$$

Let's write this for  $s_2$  and noting the fact that we either stay at  $s_2$  or transition to  $s_1$  with the given probabilities and  $V^{\pi_L}(s_1) = 0$  (given).

$$V^{\pi_L}(s_2) = 0.9(-1 + 1 \cdot V^{\pi_L}(s_1)) + 0.1(-1 + 1 \cdot V^{\pi_L}(s_2))$$

This is a simple equation, we solve for  $V^{\pi_L}(s_2) = 8/9$ . Repeat for  $V^{\pi_L}(s_3)$  and plug-in  $V^{\pi_L}(s_2)$ .

$$V^{\pi_L}(s_3) = 0.9(-1 + 1 \cdot V^{\pi_L}(s_2)) + 0.1(-1 + 1 \cdot V^{\pi_L}(s_3)) \Rightarrow V^{\pi_L}(s_3) = -2/9$$

Repeat for  $V^{\pi_L}(s_4)$  and plug-in  $V^{\pi_L}(s_3)$ .

$$V^{\pi_L}(s_4) = 0.9(-1 + 1 \cdot V^{\pi_L}(s_3)) + 0.1(-1 + 1 \cdot V^{\pi_L}(s_4)) \Rightarrow V^{\pi_L}(s_4) = -12/9$$

- (b) (4 points) Policy Improvement: What is the action you would chose for  $s_3$  based on the current state values? Do you expect this to change if we continue with policy iteration? We apply policy extraction/improvement step from the policy iteration algorithm:

$$\pi'(s_3) = \arg \max_a \sum_{s'} P(s'|s, a)(R(s') + \gamma V^{\pi}(s')).$$


We already have the  $L$  action values as  $-2/9$ . We need to calculate the value for  $R$ :

$$0.9(-1 + 1 \cdot V^{\pi}(s_4)) + 0.1(-1 + 1 \cdot V^{\pi}(s_3)) = -20/9$$

Based on this, the  $L$  action would be lead to a higher value. However, by looking at the grid world, we can see that the  $R$  action is better for  $s_3$  and if we continue, we would reach this.

2 points for values, 1 point for correct action and 1 point for reasoning.

9. (15 points) Consider the grid-world environment below.

3		-80	+100
2			
1	+30	-100	+80
	1	2	3

You want the agent to learn the optimal policy and define a MDP with the following components:

- States: Grid Cells  $s = (x, y)$
  - Actions: Up( $U$ ), Down( $D$ ), Left( $L$ ), Right( $R$ )
  - Transition Model: The actions take the agent **deterministically** to neighboring cells based on the action. (e.g.  $(1, 3), D \rightarrow (1, 2)$ )
  - Start State:  $(1, 3)$  (as given below)
  - Terminal States: All the states with numbers
  - Reward Function: 0 for empty cells and the given numbers for the other cells. The agent receives a reward when transitioning into the cells. (e.g.  $(2, 2), D \rightarrow (2, 1)$  and receives -100 reward)
  - Discount Factor:  $\gamma = 0.5$
- (a) (3 points) What is the value of the optimal value function  $V^*(s)$  at the following states? Show your work to get full points. Remember that actions are deterministic!

$$V^*((3, 2)) = \underline{100} \quad V^*((2, 2)) = \underline{50} \quad V^*((1, 3)) = \underline{15}$$

We are asked optimal values. Since the actions are deterministic, this one is easy. For  $(2, 2)$  and  $(1, 3)$ , follow the shortest path to +100 and discount accordingly. For  $(3, 2)$ , compare the discounted reward of +30 ( $30/2 = 15$ ) and +100 ( $100/2^3 = 12.5$ ) and pick the larger one.

(The question continues in the next page)

- (b) (4.5 points) You are given the following transitions from multiple episodes of the agent in this environment. Each line of an episode contains a tuple in the  $(s, a, r, s')$  format.

Episode 1	Episode 2	Episode 3
$(1, 3), D, 0, (1, 2)$	$(1, 3), D, 0, (1, 2)$	$(1, 3), D, 0, (1, 2)$
$(1, 2), R, 0, (2, 2)$	$(1, 2), R, 0, (2, 2)$	$(1, 2), R, 0, (2, 2)$
$(2, 2), D, -100, (2, 1)$	$(2, 2), R, 0, (3, 2)$	$(2, 2), R, 0, (3, 2)$
	$(3, 2), U, +100, (3, 3)$	$(3, 2), D, +80, (3, 1)$

You decide to use Q-Learning to calculate the Q-values with the learning rate  $\alpha = 0.5$ . What are the following Q-value after the above three episodes as calculated by Q-learning? Assume that all the Q-values are initialized to 0 and that the Q-values are updated based on the order seen in the episode. Show your work to get full points but you can ignore updates that do not change anything (i.e. keep things at 0).

$$Q((3, 2), U) = \underline{50} \quad Q((1, 3), D) = \underline{0} \quad Q((2, 2), R) = \underline{12.5}$$

We are going to apply use Q-Learning:

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

Some of you got confused with the “max” part and asked me which action to take. Q-learning uses the argmax action so you do not need to know the next one. The method that uses the next action as well is called “SARSA” learning.

First Episode: First two-steps are all zeros so no updates. With terminal state q-values as 0.

$$\begin{aligned} \text{Last step } Q((2, 2), D) &\leftarrow Q((2, 2), D) + 0.5 \cdot (-100 + 0.5 \max_{a'} (Q(s', a')) - Q((2, 2), D)) \\ &= Q((2, 2), D) \leftarrow 0 + 0.5 \cdot (-100 + 0.5 \cdot 0 - 0) = -50 \end{aligned}$$

Second Episode: Again only the last step matters (since the max for  $(2, 2)$  will be 0).

$$\text{Last step } Q((3, 2), U) = 0 + 0.5 \cdot (100 + 0.5 \cdot 0 - 0) = 50$$

Third Episode: Only the third step (since max Q-value for  $(3, 2)$  is not zero but is 50) and forth step results in an update:

$$\text{Third step } Q((2, 2), R) = 0 + 0.5 \cdot (0 + 0.5 \cdot 50 - 0) = 12.5$$

At this point next step doesn't matter since it is not asked.

$$\text{Last step } Q((3, 2), D) = 0 + 0.5 \cdot (80 + 0.5 \cdot 0 - 0) = 40$$

(c) (7.5 points) Consider a feature based representation of the Q-value function as follows:

$$Q_f(s, a) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(a)$$

where

- $f_1(s)$  is  $x$  (first) coordinate minus 2:  $f_1(s) = x - 2$
- $f_2(s)$  is  $y$  (second) coordinate minus 2:  $f_2(s) = y - 2$
- $f_3(a)$  given as:  $f_3(U) = 1, f_3(D) = -1, f_3(L) = -2, f_3(R) = 2$  and for the terminal states  $f_3(\cdot) = 0$

Using the episode 1 of part (b) and with the initial weights  $w_1 = w_2 = 0, w_3 = 50$ , calculate/update the weights. Show your work to get full points but you can ignore updates that do not change anything (i.e. keep things at 0). Note that you need to update the weights 3 times and for partial, show the intermediate values.

$$w_1 = \underline{25}$$

$$w_2 = \underline{-50}$$

$$w_3 = \underline{12.5}$$

We are going to apply use approximate Q-Learning:

$$w_i \leftarrow w_i + \alpha(r + \gamma \max_{a'}(Q(s', a')) - Q(s, a)) f_i(s, a) = w_i + \alpha \cdot \text{difference} \cdot f_i(s, a)$$

For this we don't need any initial Q-values since we already have the equation given to us. We will do the updates for the first transition in detail:

For  $s = (1, 3), a = D, r = 0, s' = (1, 2)$

For  $s$ :  $f_1(s) = 1 - 2 = -1, f_2(s) = 3 - 2 = 1, f_3(a) = -1$

Then  $Q((1, 3), D) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(a) = 0 \cdot (-1) + 0 \cdot 50 \cdot (-1) = 0 + 0 - 50 = -50$

For  $s'$ :  $f_1(s') = 1 - 2 = -1, f_2(s') = 2 - 2 = 0$ . We need to take the argmax for  $a'$ . Looking at the weights, other features and the equation, we can see that  $a' = R$  ( $w_3 > 0 \rightarrow R, w_3 < 0 \rightarrow L$ ) so  $f_3(a') = 2$

Then  $\max_{a'}(Q(s', a')) = 0 \cdot (-1) + 0 \cdot (0) + 50 \cdot 2 = 100$ . We plug everything in

Difference:  $r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (0 + 0.5 \cdot 100 - (-50)) = 100$

$$w_1 \leftarrow 0.0 + 0.5 \cdot 100 \cdot -1 = -50$$

$$w_2 \leftarrow 0.0 + 0.5 \cdot 100 \cdot 1 = 50$$

$$w_3 \leftarrow 50.0 + 0.5 \cdot 100 \cdot (-1) = 0$$

For the second transition  $s = (1, 2), a = R, r = 0, s' = (2, 2)$ :

$f_1((1, 2)) = -1, f_2((1, 2)) = 0, f_3(R) = 2, Q((1, 2), R) = -50 \cdot -1 + 50 \cdot 0 + 0 \cdot 2 = 50$

$f_1((2, 2)) = 0, f_2((2, 2)) = 0$ . Since  $w_3 = 0$ , action not important.  $Q((2, 2), \cdot) = 0$

Difference:  $r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (0 + 0.5 \cdot 0 - 50) = -50$

$$w_1 \leftarrow -50.0 + 0.5(-50.0)(-1) = -25.0$$

$$w_2 \leftarrow 50.0 + 0.5(-50.0)0 = 50.0$$

$$w_3 \leftarrow 0.0 + 0.5(-50.0)2 = -50.0$$

For the third transition  $s = (2, 2), a = D, r = -100, s' = (2, 1)$ :

$f_1((2, 2)) = 0, f_2((2, 2)) = 0, f_3(D) = -1, Q((2, 2), D) = 50$

$f_1((2, 1)) = 0, f_2((2, 1)) = -1, f_3(L) = -2, Q((2, 1), L) = 50$

Difference:  $r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (-100 + 0.5 \cdot 50 - 50) = -125$

$$w_1 \leftarrow -25.0 + 0.5(-125)0 = -25.0$$

$$w_2 \leftarrow 50.0 + 0.5(-125)0 = 50.0$$

$$w_3 \leftarrow -50.0 + 0.5(-125)(-1) = 12.5$$

SARSA version is on the next page which gets half points:

For the first transition  $s = (1, 3), a = D, r = 0, s' = (1, 2), a' = R$ :

$$f_1((1, 2)) = -1, f_2((1, 2)) = 1, f_3(D) = -1, Q((1, 3), D) = -50$$

$$f_1((2, 2)) = 0, f_2((2, 2)) = 0, f_3(R) = 2, Q((2, 2), R) = 100$$

$$\text{Difference: } r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (0 + 0.5 \cdot 100 - (-50)) = 100$$

$$w_1 \leftarrow 0.0 + 0.5 \cdot 100 \cdot -1 = -50$$

$$w_2 \leftarrow 0.0 + 0.5 \cdot 100 \cdot 1 = 50$$

$$w_3 \leftarrow 50.0 + 0.5 \cdot 100 \cdot (-1) = 0$$

For the second transition  $s = (1, 2), a = R, r = 0, s' = (2, 2), a' = D$ :

$$f_1((1, 2)) = -1, f_2((1, 2)) = 0, f_3(R) = 2, Q((1, 2), R) = 50$$

$$f_1((2, 2)) = 0, f_2((2, 2)) = 0, f_3(D) = -1, Q((2, 2), D) = 0$$

$$\text{Difference: } r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (0 + 0.5 \cdot 0 - 50) = -50$$

$$w_1 \leftarrow -50.0 + 0.5(-50.0)(-1) = -25.0$$

$$w_2 \leftarrow 50.0 + 0.5(-50.0)0 = 50.0$$

$$w_3 \leftarrow 0.0 + 0.5(-50.0)2 = -50.0$$

For the third transition  $s = (2, 2), a = D, r = -100, s' = (2, 2)$ .  $a'$  doesn't matter as  $s'$  is terminal:

$$f_1((2, 2)) = 0, f_2((2, 2)) = 0, f_3(D) = -1, Q((1, 2), R) = 50$$

$$f_1((2, 1)) = 0, f_2((2, 1)) = -1, f_3(\cdot) = 0, Q((2, 2), D) = -50$$

$$\text{Difference: } r + \gamma \max_{a'}(Q(s', a')) - Q(s, a) = (-100 + 0.5 \cdot (-50) - 50) = -175$$

$$w_1 \leftarrow -25.0 + 0.5(-175)0 = -25.0$$

$$w_2 \leftarrow 50.0 + 0.5(-175)0 = 50.0$$

$$w_3 \leftarrow -50.0 + 0.5(-175)(-1) = 37.5$$