Robert Sanchez

Bee Chang

IS 160

Dr. Stephen Choi

Lab 11 Reinforcement Learning/Markov Decision Process

Checkers:
Agent: the black and red pieces moving on the board
Action: the act of each piece moving once per turn
Environment: the board and the black and red pieces
State: after each turn or movement the system would capture the endturn state
Reward: there would probably be a point system around reducing the amount of pieces the opposing color has and maintaining their own.

States:

1. Black moves first. Moving within the defined constraints
    a. The environment changes due to discovery and new agent placement
2. Red moves: each movement has no points just yet
3. Black moves: as they approach each other they get closer to finding the points
4. Red moves: hypothetically eats a black piece
    a. Red finds the first reward; and black finds the first punishment
5. Black moves: find no red near
6. Red moves: finds no black that are 'edible'
7. Black moves: eats the red
    a. Black gains its first reward
8. Red moves: finds no black pieces
9. Black moves: finds no red pieces
10. Red moves: finds no black pieces

Its hard to say whether or not our state value are too small and that we should capture it by a game by game basis and not a move by move. But theres always a trial and error, and in this case i think it just might be an error.

1		●		●		●		●
2	●		●		●		●	
3		●		●		●		●
4	.12		.33		.84		.58	
5		.28		.86		.63		.21
6	○		○		○		○	
7		○		○		○		○
8	○		○		○		○	

A B C D E F G H

State = R(F,3) + γ [.8(E,4) + 0.1(G,4)] = .62876

1		●		●		●		●
2	●		●		●		●	
3		●		●		.84		●
4	.12		.33		●		.58	
5		.28		.63		.87		.21
6	○		○		○		○	
7		○		○		○		○
8	○		○		○		○	

A

B C D E F G H

State = R(E,6) + γ [.8(F,5) + 0.1(D,5))] = .86108

1		●		●		●		●
2	●		●		●		●	
3		●		●		.84		●
4	.12		.33		●		.58	
5		.28		.86		○		.21
6	○		○		.55		○	
7		○		○		○		○
8	○		○		○		○	

A

B C D E F G H

State = R(G,2) + γ [1(F,3)] = .07812

1		●		●		●		●
2	●		●		●		.33	
3		●		●		●		●
4	.77		.55		●		.58	
5		.84		.12		○		.67
6	○		○		.55		○	
7		○		○		○		○
8	○		○		○		○	

A

B C D E F G H

State = R(C,4) + γ [.8(B,5) + .1(D,5)] = .78364

1		●		●		●		●
2	●		●		●			
3		●		●		●		●
4	.79		.33		●		.58	
5		○		.77		○		.21
6	○		.89		.55		○	
7		○		○		○		○
8	○		○		○		○	

A

B C D E F G H

State = R(B,3) + γ [.8(A,4) + .1(C,4)] = .65635

1		●		●		●		●
2	●		●		●		.09	
3		.21		●		●		●
4	●		.19		●		.58	
5		○		.55		○		.21
6	○		.87		.77		○	
7		○		○		○		○
8	○		○		○		○	

A

B C D E F G H

State = R(F,7) + γ [1(E,6)] = 1.0182

1		●		●		●		●
2	●		●		●		.09	
3		.21		●		●		●
4	●		.19		●		.58	
5		○		.55		○		.21
6	○		.87		○		○	
7		○		○		.33		○
8	○		○		○		○	

A

B C D E F G H

State = R(A,4) + γ [1(C,6)] = 1.264

1		●		●		●		●
2	●		●		●		.09	
3		.21		●		●		●
4			.19		●		.58	
5		.87		.91		○		.21
6	○		●		○		○	
7		○		○		.33		○
8	○		○		○		○	

A

B C D E F G H

State = R(B,7) + γ [1(D,5)] = 1.037

1		●		●		●		●
2	●		●		●		.09	
3		.21		●		●		●
4			.19		●		.58	
5		.87		○		○		.21
6	○		.93		○		○	
7				○		.33		○
8	○		○		○		○	

A
B
C D E F G H

State = R(E,4) + γ [1(C,6)] = 1.1954

1		●		●		●		●
2	●		●		●		.09	
3		.21		●		●		●
4			.19		.07		.58	
5		1		.77		○		.67
6	○		●		○		○	
7				○		.33		○
8	○		○		○		○	

A
B C D E F G H

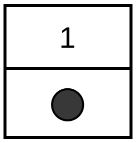
State = R(D,7) + γ [1(B,5)] = 1.3148

Maze:
Agent: the entity moving inside the maze
Action: the act of moving around inside the maze
Environment: the maze and entity
State: after each movement or after each unsuccessful trial period
Reward: The reward would need to incentivise movement towards an exit and away from the initial starting location. So points for exploring and point for distance away from starting. I could already see problems with point reduction everytime it needs to turn back and make a different turn since it doesn't want to lose points. Then there's the possibility that maybe there is a path that maximizes more points by just going through to the other corner of the maze instead of exiting.

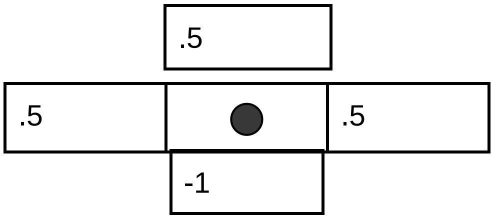
States:

1. The ai does not move
    a. It does not know what moving is
2. The ai moves in a straight line into the wall
    a. In this case in order to reset positions/states walls are now lethal and touching them rests the position
3. The ai probably has not learnt to avoid the wall just yet as it gains a few points every times it just moves forward
4. The ai learns to turn into a different corridor
5. Ai learns that turning generates more points and turns often
6. Ai traps itself into the same deadend
    a. Repeats the same exact movements
7. Ai learns a new path with more point generation
8. Ai reaches another deadend
9. Ai learns that walls are the biggest punishment by taking away all its points
10. Ai learns to identify walls and turn back away from them

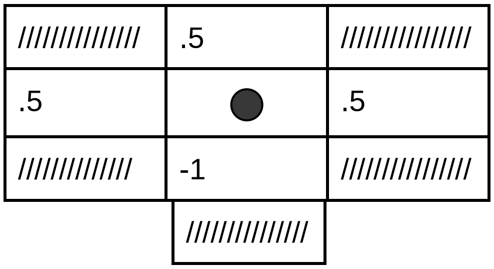
In this case the states are on a much bigger scale and would take less storage space recalling each state. However, depending on the complexity of a maze this type of learning is far too slow even with my generous learning pace that i gave it. A better solution would be to implement multiple entities roaming the maze in one generation/iteration.



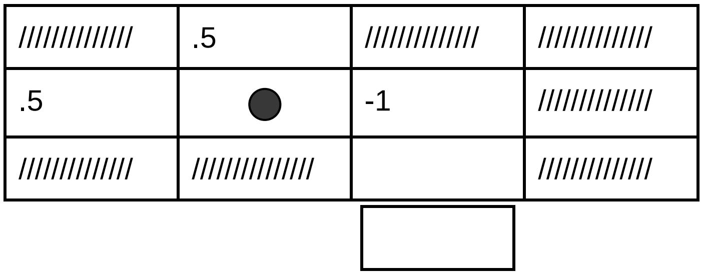
State = $R(0,0) + \gamma\ [1(0,1)] = 1$



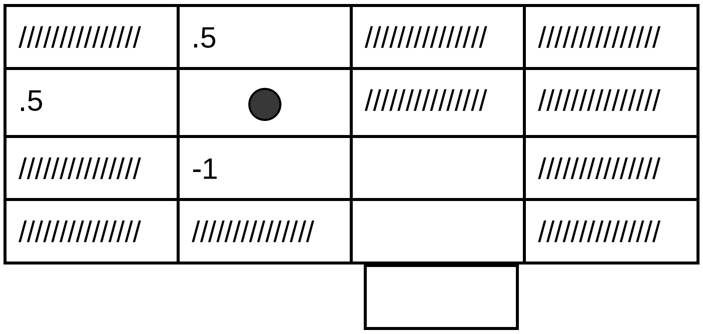
State = R(0,1) + γ [.33(1,1) + .33(-1,1) +.33(0,2)] = .93065



State = R(0,2) + γ [.33(1,2) + .33(-1,2) +.33(0,3)] = .93065



State = R(-1,2) + γ [.5(-2,-2) + .5(-1,3)] = 1.5



State = R(-1,3) + γ [.5(-2,-3) + .5(-1,4)] = 1.5

//////////////	.5	//////////////	//////////////
.5	●	.5	//////////////
//////////////	-1	//////////////	//////////////
//////////////			//////////////
//////////////	//////////////		//////////////

□

State = R(-1,4) + γ [.33(-2,4) + .33(-0,4) +.33(-1,5)] = .93065

//////////////	1	//////////////	//////////////
//////////////	●	//////////////	//////////////
//////////////	-1	//////////////	//////////////
//////////////		//////////////	//////////////
//////////////		□	//////////////
//////////////	//////////////		//////////////

State = R(-1,5) + γ [1(-1,6)] = 1

//////////////	//////////////	//////////////	//////////////
//////////////	●	1	//////////////
//////////////	-1	//////////////	//////////////
//////////////		//////////////	//////////////
//////////////		//////////////	//////////////
//////////////			//////////////
//////////////	//////////////		//////////////

□

State = R(-1,6) + γ [1(0,6)] = 1

//////////////	//////////////	.5	//////////////
//////////////	-1	●	.5
//////////////		//////////////	//////////////
//////////////		//////////////	//////////////
//////////////		//////////////	//////////////
//////////////			//////////////
//////////////	//////////////		//////////////

//////////////

State = R(0,6) + γ [.5(0,7) + .5(1,6)] = 1.5

//////////////	//////////////	//////////////	.5	
//////////////		-1	●	.5
//////////////		//////////////	//////////////	//////////////
//////////////		//////////////	//////////////	//////////////
//////////////			//////////////	//////////////
//////////////	//////////////	//////////////	//////////////	//////////////
////////////////// ///////////	//////////////		//////////////	//////////////

State = R(1,6) + γ
[.5(1,7) + .5(2,6)] = 1.5

Crow/Raven Training:
Agent: a crow or raven (any other smart bird), human(s)
Action: flying, perching, observing, picking-up trash, depositing trash,
Environment: A whole city, crows, treat dispenser
State: The days counting after placing the treat dispenser
Reward: a tasty treat

States:

1. Day 1: Trash litters the park, crows flying by, and a newly installed treat dispenser
2. Day 2: human displaying how to use the treat dispenser
3. Day 3: curious crows comes to see the machine
4. Day 4: some crows tries testing rocks
    a. Machine is designed to only take trash
5. Day 5: a crow manages to pick up a piece of trash and returns it for some treat
6. Day 6: crows are better able to identify trash versus other objects
    a. The flow of treats/rewards increases
7. Day 7: crows are smart enough to exploit the system and steal not-yet-trash from the hands of humans
8. Day 8: There is noticeably less trash littering the park, but also because there are less people. Both because of the crows
9. Day 9: the treat dispenser is removed
10. Day 10: the people return and so did the trash, and some of the crow behavior of stealing remains

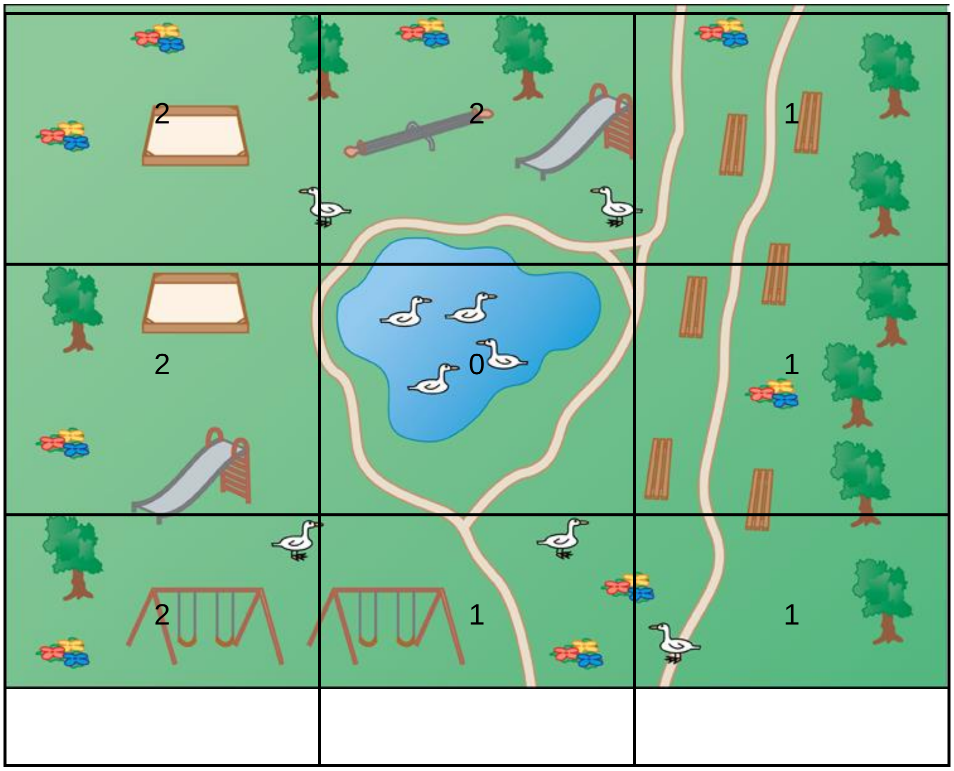
I don’t know if this is a good example that you’re looking for, but it was based **loosely** on an experiment that researchers were doing to reduce litter through nature. The crows are the agent whose natural curiosity and bright mind was utilized to minimize the need to facilitate direct teaching. Kind of like a reinforcement learning ai. I think that it has some of the quirks that an ai would have as well. Namely, the ability to even see that trash is a very loose term and exploit it for a quicker reward farm. Sometimes in an effort to not be punished, refuses to do anything.

Bellman equation:
$Q(s,a)=R(s,a)+\gamma\times\max(Q(s',a'))$

In this example discount factor for the bellman equation will be 0.5

Values for this example:

- Starting Point: 0
- Pathway Areas: 1 each
- Playground Parks: 2 each
- The Pond: 0 each
- Trash Return: 10



State	Action	Q-value calculation
Starting Point	Trash Return	0+0.5×10=5
Starting Point	Trash Return	0+0.5×10=5
Starting Point	Trash Return	0+0.5×10=5
Playground	Trash Return	2+0.5×10=7
Playground	Trash Return	2+0.5×10=7
Walkway	Trash Return	1+0.5×10=6
Playground	Trash Return	2+0.5×10=7
The Pond	Trash Return	0+0.5×10=5
Walkway	Trash Return	1+0.5×10=6
Playground	Trash Return	2+0.5×10=7
Walkway	Trash Return	1+0.5×10=6
Walkway	Trash Return	1+0.5×10=6

Drone Delivery Service
Agent: A delivery drone optimizing its route to deliver packages.
Action: Delivering packages in a city
Environment: A city with different neighborhoods and package delivery locations.
State: specific locations or conditions within the city environment where the drone must navigate and make decisions
Reward: encourages the drone to learn and select the most beneficial route to ensure a successful delivery

States:

1. Starting Point: The drone begins its delivery route.
2. Residential Area A: A neighborhood in the city.
3. Residential Area B: Another neighborhood in the city.
4. Office Park A: Business district with delivery locations.
5. Office Park B: Another business district with delivery locations.
6. Package Delivery 1: Specific delivery location in Office Park A.
7. Package Delivery 2: Specific delivery location in Office Park B.
8. Empty Space 1: Area with no delivery locations in Residential Area A.
    - The drone can use this opportunity to scan for alternative routes
9. Empty Space 2: Area with no delivery locations in Residential Area B.
    - The drone can use this opportunity to scan for alternative routes
10. Final Delivery Point: Last delivery location before completion.

In this example, the delivery drone navigates through different states within the city, making decisions at each state to optimize its delivery route and maximize the total reward by efficiently delivering packages while learning the most valuable actions in various situations.

Lab 11
Bellman equation:
$Q(s,a)=R(s,a)+\gamma\times\max(Q(s',a'))$

In this example discount factor for the bellman equation will be 0.9

Values for this example:

- Starting Point: 0
- Residential Areas: 1 each
- Office Parks: 2 each
- Package Deliveries: 5 each
- Empty Spaces: 0.5 each
- Final Delivery Point: 10

State	Action	Q-value calculation
Starting Point	Residential A	0+0.9×1=0.9
Starting Point	Residential B	0+0.9×1=0.9
Residential A	Office Park A	1+0.9×2=2.8
Residential A	Empty Space 1	1+0.9×0.5=1.45
Residential B	Office Park B	1+0.9×2=2.8
Residential B	Empty Space 2	1+0.9×0.5=1.45
Office Park A	Package Delivery 1	2+0.9×5=6.5
Office Park B	Package Delivery 2	2+0.9×5=6.5
Package Delivery 1	Final Delivery Point	5+0.9×10=14.5
Package Delivery 2	Final Delivery Point	5+0.9×10=14.5
Empty Space 1	Residential A	0.5+0.9×1=1.4
Empty Space 2	Residential B	0.5+0.9×1=1.4

Patient treatment in the healthcare industry
Agent: A healthcare AI system managing patient treatment
Action: Initiate the correct treatments/procedures depending on the patient's condition
Environment: A hospital setting with various patient conditions.
State: phases or conditions a patient might encounter
Reward: optimize patient care and improve health outcomes

States:

1. Admission: Initial state when a patient is admitted.
2. Patient Stable: Patient's condition is stable and recovering.
3. Patient Critical: Patient is in a critical condition.
4. Post-Treatment: Patient post-treatment but needs monitoring.
5. Discharge: Patient is ready for discharge.
6. Post-Discharge Care: Patient's status after leaving the hospital.
7. Readmission: Scenario where a patient is readmitted.
8. Patient Improving: Patient's health condition is improving.
9. Patient Deteriorating: Patient's health condition is deteriorating.
10. Critical Condition: Severe state of patient health.

This example of this reinforcement learning is to help analyze the patients that enter the hospital and help the AI figure out the severity of the patients health. Depending on the severity, this can help choose the most appropriate action for each state that can improve health or speed up a recovery.

Lab 11
Bellman equation:
$Q(s,a)=R(s,a)+\gamma\times\max(Q(s',a'))$

In this example discount factor for the bellman equation will be 0.9

Values for this example:

- Admission: 0
- Patient Stable: 1
- Patient Critical: -5 (as it's a critical state)
- Post-Treatment: 2
- Discharge: 3
- Post-Discharge Care: 1
- Readmission: -3
- Patient Improving: 1
- Patient Deteriorating: -1
- Critical Condition: -5

State	Action	Q-value calculation
Admission	Patient Stable	0+0.9×1=0.9
Admission	Patient Critical	0+0.9×(−5)=−4.5
Patient Stable	Post-Treatment	1+0.9×2=2.8
Patient Stable	Discharge	1+0.9×3=3.7
Patient Stable	Patient Improving	1+0.9×1=1.9
Patient Critical	Post-Treatment	(−5)+0.9×2=−3.1
Patient Critical	Readmission	(−5)+0.9×(−3)=−7.7
Post-Treatment	Discharge	2+0.9×3=4.7
Post-Treatment	Post-Discharge Care	2+0.9×1=2.9
Discharge	Post-Discharge Care	3+0.9×1=3.9
Post-Discharge Care	Readmission	1+0.9×(−3)=−1.7
Post-Discharge Care	Patient Deteriorating	1+0.9×(−1)=0.1
Post-Discharge Care	Patient Improving	1+0.9×1=1.9
Readmission	Patient Critical	(−3)+0.9×(−5)=−7.5
Patient Improving	Patient Stable	1+0.9×1=1.9
Patient Improving	Post-Treatment	1+0.9×2=2.8
Patient Deteriorating	Patient Critical	(−1)+0.9×(−5)=−5.5
Patient Deteriorating	Critical Condition	(−1)+0.9×(−5)=−5.5

Managing a restaurant
Agent: An AI system managing a restaurant kitchen
Action: specific tasks being performed in the kitchen
Environment: A restaurant kitchen with various cooking stations.
State: different stages or phases of kitchen operations
Reward: delivering satisfying customer experiences. This tells the model that these kitchen operations are working to satisfy the customer in a quick and timely manner

States:

1. Order Received: Initial state upon receiving a customer's order.
2. Ingredient Preparation: State when ingredients are being readied for cooking.
3. Cooking in Progress: State where the dish is being prepared.
4. Dish Ready for Plating: The dish is ready to be plated for serving.
5. Plating and Garnishing: The final stage before serving the dish.
6. Serving Completed: Dish has been served to the customer.
7. Customer Satisfied: State indicating customer satisfaction.
8. Customer Unsatisfied: Reflecting an unsatisfactory customer experience.
9. Busy Period: High demand and activity in the kitchen.
10. Idle Period: Low activity and demand in the kitchen.

This reinforcement learning environment focuses on optimizing kitchen operations in a restaurant to deliver exceptional customer service. The AI agent adapts its actions in accordance with various states to maximize positive rewards, ensuring efficient operations and a superior dining experience for customers

Lab 11
Bellman equation:
$Q(s,a)=R(s,a)+\gamma\times\max(Q(s',a'))$

In this example discount factor for the bellman equation will be 0.8

Values for this example:

- Order Received: 0
- Ingredient Preparation: 1
- Cooking in Progress: 2
- Dish Ready for Plating: 3
- Plating and Garnishing: 3.5
- Serving Completed: 4
- Customer Satisfied: 5
- Customer Unsatisfied: -5
- Busy Period: 1
- Idle Period: -1

State	Action	Q-value calculation
Order Received	Ingredient Preparation	0+0.8×1=0.8
Ingredient Preparation	Cooking in Progress	1+0.8×2=2.6
Cooking in Progress	Dish Ready for Plating	2+0.8×3=4.4
Dish Ready for Plating	Plating and Garnishing	3+0.8×3.5=5.8
Plating and Garnishing	Serving Completed	3.5+0.8×4=6.3
Serving Completed	Customer Satisfied	4+0.8×5=8
Customer Satisfied	(Final Destination - No action)	5 (final destination)
Order Received	Busy Period	0+0.8×1=0.8
Order Received	Idle Period	0+0.8×(−1)=−0.8
Ingredient Preparation	Busy Period	1+0.8×1=1.8
Ingredient Preparation	Idle Period	1+0.8×(−1)=0.2
Cooking in Progress	Busy Period	2+0.8×1=2.8
Cooking in Progress	Idle Period	2+0.8×(−1)=1.2
Dish Ready for Plating	Busy Period	3+0.8×1=3.8
Dish Ready for Plating	Idle Period	3+0.8×(−1)=2.2
Plating and Garnishing	Busy Period	3.5+0.8×1=4.3
Plating and Garnishing	Idle Period	3.5+0.8×(−1)=2.7
Serving Completed	Busy Period	4+0.8×1=4.8
Serving Completed	Idle Period	4+0.8×(−1)=3.2
Customer Satisfied	Busy Period	5 (final destination)
Customer Satisfied	Idle Period	5 (final destination)