Assignment-Like Assessment 5: Reinforcement Learning and Fake News

CS486/686 – Winter 2020

Out: April 3rd, 2020
Due: April 15th, 2020 5pm

Submit your solutions to this assessment via LEARN (CS486 site) in the Assessment 5 Dropbox folder.
No late assessments will be accepted

1. **[50 pts]** Reinforcement learning: In this problem, you will study the emergence of communication in a two-agent reinforcement learning problem.[1] One agent, the *receiver*, is in a 5×5 grid-world (with walls) and starts at the center location. A prize is hidden (from the receiver) somewhere on the grid. The prize is placed randomly but cannot be in a wall location, and is never placed at the receiver's starting position. The second agent, the *sender*, knows the location of the prize only, and can send the receiver a single message as one of a set of N symbols. The sender and the receiver each get the prize (a reward of 1.0 each) upon the receiver *entering* the cell that contains the prize. After receiving the sender's signal, the receiver acts until it finds the reward or is terminated.[2] The receiver is terminated at each step with some fixed probability, δ. These two agents must somehow figure out a signaling mechanism (they must build an agreed-upon language) that allows the receiver to find the prize most quickly.

   Implement two Q-learning agents, one for the receiver, and one for the sender. The sender agent's states are the location of the prize on the grid. Its actions are the possible symbols it can send. The receiver agent's states are tuples giving the received signal and the current location (which is fully observable [3]). The receiver can move up, down, left or right. An action that leads into a wall or out of the grid leaves the agent where it is.

   The Q-functions for these agents can simply be implemented as tables or matrices that you fill in with a nested loop.[4] The outer loop consists of the sender taking an action (starting an "episode"), while the inner loop consists of the receiver acting until it is terminated. When terminated, the receiver's learning episode ends. The receiver has a δ chance of being terminated after each action it takes, and receives a reward of 0 if this happens. It is also terminated (and the episode ends) after moving into the prize's grid cell, but receives a reward of 1.0.[5] The prize is placed randomly at each episode, never at a wall, and never at the start location. The receiver is not penalized for moving into a wall (but the move is ineffective).

   You will train the Q-learning agents over a series of N_{ep} *episodes*, each consisting of a sender action (message), followed by a series of receiver actions (moves until termination). Each agent updates its Q-function after each of its iterations. Your Q-learning agents should use ϵ-greedy exploration during training, and should switch to greedy actions when testing (switch to $\epsilon = 0$ for testing). Use a learning rate α which starts at $\alpha_0 = 0.9$ and decreases linearly down to $\alpha_f = 0.01$ over the episodes until N_{ep} episodes.[6] Use a discount factor of $\gamma = 0.95$ and a

[1] This problem is based on the following paper: Ivana Kajić, Eser Aygün and Doina Precup. *Learning to cooperate: Emergent communication in multi-agent navigation* arXiv:2004.01097, 2020 https://arxiv.org/abs/2004.01097

[2] You could also construct a related problem where the receiver is penalized for each step it takes without receiving the reward. This may make learning the shortest path solution easier.

[3] You may also want to consider this problem in the partially observable case.

[4] You can use any other Q-function representation if you prefer, e.g. a deep neural network, but a table will suffice for this example. To simplify the implementation, you can include a "dummy" representation for the wall states, i.e., the Q-table can have rows that are never updated. However, you may have a fancier representation that doesn't waste the wall space.

[5] You may feel a certain sadness for the receiver in this case, but you must understand that your receiver re-spawns instantly when terminated, and keeps its reward, and will continue to do so indefinitely!

[6] That is, decrease the learning rate by $\frac{\alpha_0 - \alpha_f}{N_{ep}}$ each step taken by the receiver.

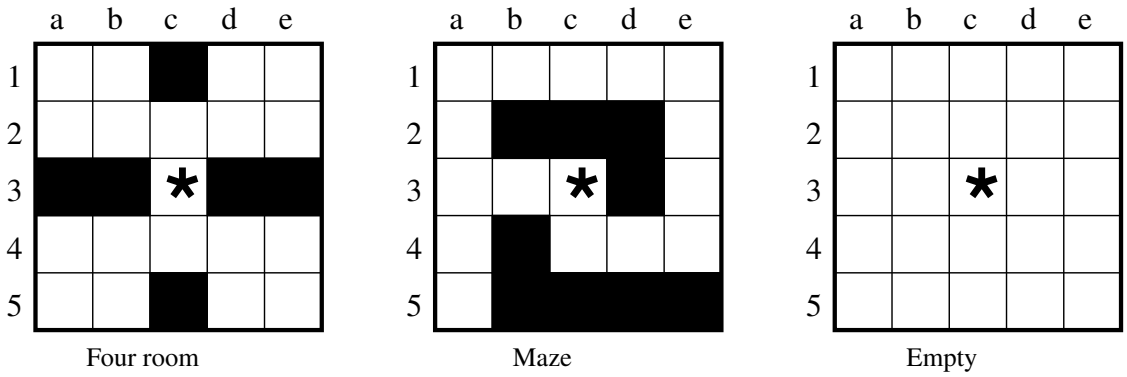


Figure 1: Domains to be used. The robot start location is shown with a "*".

termination probabilty of $\delta = 1 - \gamma$.

You will use the grids shown in Figure 1. The start position is always the center (marked "*") and the prize location is randomly selected at each episode, but is never at the start location and is never in a wall location (or off the grid).

Submit a printout of your code in a single second file, and then answer the following questions.

(a) Consider an empty 5×5 grid. What is the discounted average reward for the optimal search strategy for the receiver to take if the sender has only a single signal (so the sender is useless) and the receiver is only terminated when it reaches the prize? There may be many such optimal search strategies. Note however that the receiver can only move in a single direction from a grid cell (it has a history of 0), so the optimal search strategy will not cross its own path. The first path found by depth-first search using a rule of up,left,down,right with a cycle check is an example of an optimal search strategy. Draw an optimal search strategy on 5×5 grid stating at the middle, and write the discounted average reward using $\gamma = 0.95$.

(b) Now consider the "four room" grid we introduced above in Figure 1. [7] There is no way to do this without a history of at least one in the state (which is possible). However, the historyless agent [8] can randomize, thereby getting the reward in one of four cases. Such an agent selects a random room each time, receiving the expected reward in only one of four cases. What is the receiver's discounted average reward over a series of trials without a signal in this room? Use $\gamma = 0.95$.

(c) Using the "four room" grid shown above and $N = 4$ signals, run 10 tests (each test being a set of N_{ep} episodes) for each of $N_{ep} = 10, 100, 1000, 10000, 10000, 100000$ and for each of $\epsilon = 0.01, 0.1, 0.4$. Each episode is a single update step for the sender, but is a randomized number of updates (depending on δ) for the receiver. After each test, run the trained agents for an additional 1000 episodes using $\epsilon = 0$, measuring the discounted reward obtained by either sender or receiver. Plot the average discounted reward received as a function of $\log(N_{ep})$ for each value of ϵ. Plot error bars as standard deviations over the 10 tests. Show an example of a policy for the $N = 4$ case with $N_{ep} = 100000$. Show 4 grids, one for each action with a $\rightarrow$ in each grid cell showing the direction that would be taken.

(d) Using $\epsilon = 0.1$, run the same set of tests for different N values of $N = 2, 4, 10$. Show a single graph with three lines on it with errorbars showing average discounted reward of for each N values as a function of N_{ep}.

(e) Using the "maze" grid, run the same 10 tests as above, but for each of $N = 2, 3, 5$ and report the same graph of the average discounted reward as in the last question. Use only $\epsilon = 0.1$

(f) Finally, using the "empty" grid, run a further 10 tests. Use only $\epsilon = 0.1$ and $N = 1$. Again, report the same graph.

[7] A "room" is this grid is a 2×2 square of cells in each of the four corners.

[8] A historyless agent only knows where it currenty is, and does not remember where it has been

(g) How many iterations of Q-learning to you think you will need to run to learn the optimal search strategy for the empty domain with one signal (i.e. with no sender). Use your graph from Question 1f and your result from Question 1a.

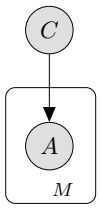
What to hand in:

- A printout of your code as a separate file.
- Question 1a: a grid showing the optimal search strategy and the average discounted reward.
- Question 1b: the average discounted reward.
- Question 1c: A graph showing average discounted reward received as a function of N_{ep} for each value of ϵ
- Question 1d: A graph showing average discounted reward received as a function of N_{ep} for each value of $N = 2, 4, 10$.
- Question 1e: A graph showing average discounted reward received as a function of N_{ep} for each value of $N = 2, 3, 5$.
- Question 1f: A graph showing average discounted reward received as a function of N_{ep} for $N = 1$.
- Question 1f: An estimate of the number of iterations of Q-learning will be needed to solve the empty room problem.

2. **[50 pts]** Fake News

Fake news is a major problem with important ramifications in today's online society. In this problem, you will design a Bayesian network classifier that can estimate the probability of a news report being true or false (fake). Consider a **single** newsworthy event that can either be true or false. This event is reported by M different news outlets, each of which publishes a report that either says the news event is true or says it is false. Each news outlet can be either trustworthy or not. All news outlets are independent of each other in both their trustworthiness and their reports (i.e. they don't read each other's reports or consider each other's trustworthiness to generate their report, they only consider the news event itself and their own trustworthiness). Suppose your prior belief is that each news outlet is trustworthy with probabilty 0.6 (and untrustworthy with probability 0.4), and that the news event is true (an event that actually happened) with probability 0.7 and false (an event that did not happen) with probability 0.3. Further, you believe that each news outlet, if trustworthy, reports news with 99% accuracy (so it reports true news as true with probability 0.99, and it reports false news as true with probability 0.01). However, if a news outlet is untrustworthy, its accuracy drops to 80% for true events and 70% for false events (so it reports true news as true with probability 0.8 and false news as true with probability 0.3).

"Plate notation" allows you to represent a set of variables in a Bayesian network in a compact way. A "plate" is drawn around a variable or set of variables, with a constant written in one corner. To remove the plate, one replicates the variables (and links if there are any) inside the plate the number of times given by the constant. For example, a naïve Bayes model with class C and M features $A_1, A_2, \ldots, A_M$ can be represented as shown on the right. Identical probability tables do not need to be replicated, so one only writes down two probability tables in this case. You don't have to use "plate" notation for this problem.



(a) Draw a Bayesian network representing the knowledge you have about the truth of a news report. Explain precisely what each of your variables means. Include all probability tables. Use plate notation to represent sets of variables.

(b) What is the probability of a news event being false (given no evidence)

(c) How likely is a news event true given that M_t news reports say its true and $M_f = M - M_t$ news reports say its false. Express your answer as a function of M_t and M_f.

(d) Suppose you read M reports about a particular news event. How many of them do you need to read saying the event is true such that you are at least 50% certain that it is not fake news? Express your answer (M_t) as a function of the total number of reports, M.

BONUS QUESTION : **5 extra marks** Actually using the Bayesian network above will be somewhat tricky in practice, as we have to determine which articles are true and which are false. That is, we would somehow have to be able to compare two news articles and determine if they were saying the same thing, or something different. Write a short (maximum 100 word) description of how you could learn this Bayesian network from data (raw news stories) automatically. You can start by imagining you have a labeled set of news sources, and go from there.