AI Planning for Autonomy

Problem Set IX: Reinforcement Learning

Recall the following description from Problem Set VII:

Consider two football-playing robots: Messi and Suarez.

They play a simple two-player cooperate game of football, and you need to write a controller for them. Each player can pass the ball or can shoot at goal.

The football game can be modelled as a discounted-reward MDP with three states: *Messi*, *Suarez* (denoting who has the ball), and *Scored* (denoting that a goal has been scored); and the following action descriptions:

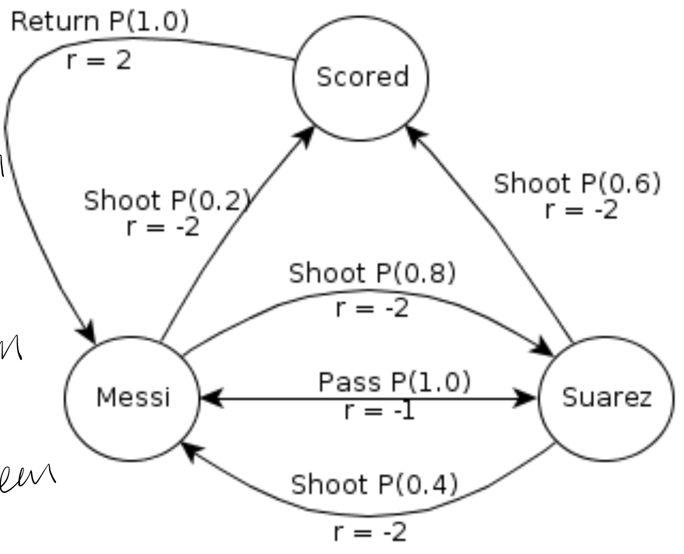
- If Messi shoots, he has 0.2 chance of scoring a goal and a 0.8 chance of the ball going to Suarez. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
- If Suarez shoots, he has 0.6 chance of scoring a goal and a 0.4 chance of the ball going to Messi. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
- If either player passes, the ball will reach its intended target with a probability of 1.0. Passing the ball incurs a cost 1 (or a reward of -1).
- If a goal is scored, the only action is to return the ball to Messi, which has a probability of 1.0 and has a reward of 2. Thus the reward for scoring is modelled by giving a reward of 2 when *leaving* the goal state.

The following diagram shows the transition probabilities and rewards:

Dynamic planning { ① value iteration ② policy iteration

↓

table → Memory problem



Tasks

1. What is the difference between Sarsa and Q-learning?
2. Assume the following Q-table, which is learnt after several episodes:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma V(s') - Q(s,a))$$

$\boxed{\text{Suarez} \xrightarrow{P} \text{Messi}}$ $Q(s,p) = Q(s,p) + \underset{\uparrow \; 0.4}{\alpha}(r(s,p) + \gamma V(\text{Messi}) - Q(s,p))$

Q-learning ↳ $\max_{a \in A} Q(s,a)$

value is defined as the states

Q-learning is defined as a set of actions

SARSA $\rightarrow Q(s, \underbrace{\pi(s)}_{a})$

Suarez $\xrightarrow{P}$ ^{a} Messi $\xrightarrow{\text{shoot}}$
s' a'

SARSA ^{s} $Q(s,P) = Q(s,P) + \alpha(r(s,P) + \gamma Q(M,S) - Q(s,P))$

$= -0.7 + 0.4(-1 + 0.9(-0.8) - (-0.7))$

$= -1.108$

State	Action Pass	Action Shoot	Action Return
Messi	-0.4	-0.8	–
Suarez	-0.7	-0.2	–
Scored	–	–	1.2

update the $Q(s', a)$ from the table.

2. In the next step of the episode, from the state 'Suarez', Suarez passes the ball to Messi. Show the Q-learning update for this action using a discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.4$.

   Note: As this is a reinforcement learning problem, assume that the transition probabilities are not accessible to your algorithm.

   Q learning $-0.7 + 0.4(-1 + 0.9 \times (-0.4) - (0.7)) = -0.964$

3. Consider again being in the state 'Suarez', Suarez passes the ball to Messi and then Messi decides to shoot. Show the SARSA update for the Pass action using a discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.4$ and assuming a' (the next action to be execute) is Shoot. Compare to the Q-learning update. What is different?

   SARSA chooses the action based on the policy. (worse or equal to Q-learning)

   However Q-learning choose the best-value action

4. Given the following trace from a historical game feed from last season:

   " Suarez passes the ball to Messi, Messi dribbles around all of his opponents, shoots and scores yet another goal! Barcelona F.C 10 - 0 Real Madrid! The ball is returned to Messi for kickoff. After he passes the ball to Suarez, the referee blew the final whistle. End of the game, the ball is taken by Messi to remember the match forever."

   Show the 3-step SARSA update for the above feed. Do you think the 1-step update is more accurate or the 3-step update? Does it indicate more steps is always better? Explain why.

Additional Tasks for Personal Study

To improve your understanding of reinforcement learning and how to work with Q functions, pick up from where you left of last week and complete the *fourth* question of Project 3 at http://ai.berkeley.edu/reinforcement.html#Q4. You should already have all the necessary files to complete this task.

Hints In order to help you complete the task, here are some useful hints:

1. The functions that you need to change:
   a) computeValueFromQValues
   b) getQValue
   c) computeActionFromQValues.
2. The files you need to take a look:
   a) util.py (Counter())
   b) featureExtractors.py
3. How to test your code:
   a) python gridworld.py -a q -k 5 -m (to get your agent to learn via keyboard input)
   b) python autograder.py -q q4 (testing by autograde)

1-step $\rightarrow$ -1.108

Q3

$S \xrightarrow{P} M$

$M \xrightarrow{\Sigma}$ scared

$S \xrightarrow{v} M$

$M \xrightarrow{P} S$.

$Q(s,p) = Q(s,p) + \alpha(r + \gamma \underbrace{Q(M,S) - Q(s,p)})$

$= -1.000864$

$Q_3 = r(s,p) + \gamma r(M,S) + \gamma^2 r(S,R)$

$+ Q^3 Q(M,p)$