Imperial College London

Coursework (Assessed): Foundations of Reinforcement Learning

Lecturer: Dr. Paul A. Bilokon

Date set: 2023.01.24
Date due: 2023.02.07

Abstract

This assignment explores the foundational ideas of reinforcement learning, exploitation and exploration, and multi-armed bandits.

The students are allowed to work on this coursework in groups of up to and including three.

1. Consider the game of noughts and crosses (also known as tic-tac-toe). The state

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}
a_{31}	a_{32}	a_{33}

can be represented by the string $s = a_{11}a_{12}a_{13}a_{21}a_{22}a_{23}a_{31}a_{32}a_{33}$, where each a_{ij}, $1 \leq i, j \leq 3$, stands for X, O, or blank.

Here is how the noughts and crosses would be approached with a method making use of a value function. First, we would set up a table of numbers, one for each possible state of the game. Each number will be the latest estimate of the probability of our winning from that state. We treat this estimate as the state's **value**, and the whole table is the learned **value function**.

State A has higher value than state B, or is considered "better" than state B, if the current estimate of the probability of our winning from A is higher than it is from B.

Assuming we always play Xs, then for all states with three Xs in a row the probability of winning is 1, because we have already won. Similarly, for all states with three Os in a row, or that are filled up, the correct probability is 0, as we cannot win from them. We set the initial values of all the other states to 0.5, representing a guess that we have a 50% chance of winning.

We then play many games against the opponent. To select our moves we examine the states that would result from each of our possible moves (one for each blank space on the board) and look up their current values in the table. Most of the times we move **greedily**, selecting the move that leads to the state with greatest value, that is, with the highest estimated probability of winning.

Occasionally, however, we select randomly from among the other moves instead. These are called **exploratory** moves because they cause us to experience states that we might otherwise never see.

While we are playing, we change the values of the states in which we find ourselves during the game. We attempt to make them more accurate estimates of the probabilities of winning. To do this, we "back up" the value of the state after each greedy move to the state before the move.

More precisely, the current value of the earlier state is updated to be closer to the value of the later state. This can be done by moving the earlier state's value a fraction of the way toward the value of the later state.

If we let S_t denote the state before the greedy move, and S_{t+1} the state after the move, then the update to the estimated value of S_t, denoted $V(S_t)$, can be written as

$$V(S_t) \leftarrow V(S_t) + \alpha \left[V(S_{t+1}) - V(S_t) \right],$$

where α is a small positive fraction called the **step-size parameter**, which influences the rate of learning.

This update rule is an example of a **temporal-difference** learning method, so called because its changes are based on a difference, $V(S_{t+1}) - V(S_t)$, between estimates at two successive times.

(a) Suppose, instead of playing against a random opponent, the reinforcement learning algorithm described above played against itself, with both sides learning. What do you think would happen in this case? Would it learn a different policy for selecting moves?

[2 marks]

(b) Suppose the reinforcement learning player was **greedy**, that is, it always played the move that brought it to the position that it rated the best. Might it learn to play better, or worse, than a nongreedy player? What problems might occur?

[2 marks]

(c) List all states that are equivalent to the state $s = a_{11}a_{12}a_{13}a_{21}a_{22}a_{23}a_{31}a_{32}a_{33}$ through symmetries.

[6 marks]

(d) How may we amend a reinforcement learning process to take advantage of these symmetries?

[1 mark]

(e) In what ways would this change improve the learning process?

[1 mark]

(f) Suppose the opponent did not take advantage of symmetries. In that case, should we? Is it true that symmetrically equivalent positions should necessarily have the same value?

[2 marks]

(g) Can you think of other ways to improve the reinforcement learning player?

[1 mark]

(h) Can you think of any better way to solve the noughts and crosses problem as posed?

[1 mark]

(i) Consider a teaching agent that is trying to teach a junior school pupil addition and subtraction. Suppose that addition is easy for the student, whereas subtraction is considerably more difficult. The reward is 1 if the student answers a question correctly and zero otherwise. What problem do you envisage with this reward structure?

[1 mark]

(j) (Programming.) Implement the proposed algorithm for noughts and crosses in Python. Train it against

i. A programmatic player that plays randomly.
ii. A programmatic player that plays randomly except when it can win on the current move — in this case that player makes the winning move.
iii. A programmatic player that plays randomly except when it can win on the current move — in this case that player makes the winning move, — and except when the opponent can win on the next move, — in this case that player makes the move required to disrupt the opponent's victory.

After appropriate training, how well does the proposed algorithm fare on each of these programmatic opponents?

[8 marks]

(k) (Programming.) Amend your implementation to take into account the ideas from the classic paper [1]. How does this impact the performance of the reinforcement learning player?

[Unassessed]

[This question carries 25 marks in total.]

2. (a) Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1, R_1 = 1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = 2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

[2 marks]

(b) From the definition

$$Q_t(a) := \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{N_t(a)},$$

where

$$N_t(A) = \sum_{i=1}^{t-1} \mathbb{1}_{A_i=a},$$

and $\mathbb{1}_{\text{predicate}}$ denotes the random variable that is 1 if the predicate is true and 0 if the predicate is false, derive the online (incremental) update equation for $Q_t(a)$ when the action a occurs:

$$Q_{t+1} = Q_t(a) + \alpha_t(a)\left[R_t - Q_t(A)\right],$$

where the step size is given by $\alpha_t(a) = \frac{1}{N_{t+1}(a)}$.

[3 marks]

(c) If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted-average of previously received rewards with a weighting different from that given by

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i.$$

What is the weighting on each prior reward for the general case, analogous to the above, in terms of the sequence of step-size parameters?

[3 marks]

(d) Consider the soft-max (i.e, Gibbs or Boltzmann) distribution used in gradient bandit algorithms:

$$\mathbb{P}\left[A_t = a\right] = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(a)}}.$$

Show that in the case of two actions, the soft-max distribution is the same as that given by the logistic, or sigmoid, function often used in statistics and artificial neural networks.

[1 mark]

(e) (Programming.) Use the 10-armed testbed introduced in class to compare with the greedy and ϵ-greedy method

- A greedy ($\epsilon = 0$) strategy with an optimistic initial value $Q_0 = 5$.
- An $\epsilon = 0.1, 0.01$ greedy strategy using upper-confidence-bound (UCB) action selection with $c = 2$.

[8 marks]

(f) (Programming.) Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like the ones demonstrated in class for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\epsilon = 0.1$ and longer runs, say of 10,000 steps.

[8 marks]

[This question carries 25 marks in total.]

References

[1] Donald Michie. Experiments on the mechanization of game-learning. Part I. Characterization of the model and its parameters. *The Computer Journal*, 6(3):232–236, November 1963.