CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55
Name: VISHNU VINOD **Roll number:** CS19B048

- This is an individual assignment. Collaborations and discussions are strictly prohibited.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- Type your solutions in the provided LaTeXtemplate file.
- **Please start early.**

1. (2 marks) [Bandit Question] Consider a N-armed slot machine task, where the rewards for each arm a_i are usually generated by a stationary distribution with mean $Q^*(a_i)$. The machine is under repair when a arm is pulled, a small fraction, ϵ, of the times a random arm is activated. What is the expected payoff for pulling arm a_i in this faulty machine?

Solution: The faulty N-armed slot machine, has the following arm that is actually pulled when the action to pull arm a_i is made:

$$\text{Pull}(a_i) = \begin{cases} a_i & \text{, with probability } 1 - \frac{(n-1)\cdot\epsilon}{n} \\ a_j \text{ , where } i \neq j & \text{, with probability } \frac{\epsilon}{n} \end{cases}$$

Thus in order to calculate the expected reward we can get the following equation:

$$\mathbb{E}_{faulty}[r(a_i)] = \left(1 - \frac{(n-1)\cdot\epsilon}{n}\right) \cdot \mathbb{E}[r(a_i)] + \frac{\epsilon}{n} \cdot \sum_{i \neq j} \mathbb{E}[r(a_j)]$$

$$\Rightarrow \boxed{\mathbb{E}_{faulty}[r(a_i)] = (1-\epsilon)\, Q^*(a_i) + \epsilon \cdot \mathbb{E}_a[Q^*(a)]} \quad (1)$$

The above equation gives us the expected value of the reward.

2. (4 marks) [Delayed reward] Consider the task of controlling a system when the control actions are delayed. The control agent takes an action on observing the state at time t. The action is applied to the system at time $t + \tau$. The agent receives a reward at each time step.

(a) (2 marks)What is an appropriate notion of return for this task?

Solution: The problem here is an example of an MDP with a constant **action delay** of τ (DDMDP - **Deterministic Delayed MDP** - Katsikopoulos and Engelbrecht, May 2003:IEEE Transactions on Automatic Control 48(4):568 - 574).

The actions are applied at a later time and the rewards due to these actions are obtained only then. Rewards obtained in intermediate steps from R_t to $R_{t+\tau-1}$ are dismissed as not being relevant (since there exists no causal connection to the action).

The return can be defined as:

$$G_t = \sum_{k=1}^{\infty} \gamma^{k-1} \cdot R_{t+\tau+k}$$

(b) (2 marks) Give the TD(0) backup equation for estimating the value function of a given policy.

Solution: The introduction of the action delay does not change the TD(0) backup equation except for which reward is used to make the backup. The reward corresponding to the action must be used and thus the equation becomes:

$$V(S_t) \leftarrow V(S_t) + \alpha \cdot (R_{t+\tau+1} + \gamma \cdot V(S_{t+1}) - V(S_t))$$

3. (5 marks) [Reward Shaping] Consider two finite MDPs M_1, M_2 having the same state set, S, the same action set, A, and respective optimal action-value functions Q_1^*, Q_2^*. (For simplicity, assume all actions are possible in all states.) Suppose that the following is true for an arbitrary function $f : S \to R$:

$$Q_2^*(s, a) = Q_1^*(s, a) - f(s)$$

for all $s \in S$ and $a \in A$.

(a) (2 marks) Show mathematically that M_1 and M_2 has same optimal policies.

Solution:

In order to mathematically, prove that M_1 and M_2 have the same optimal policies we have the optimal policy selection given below:

$$\pi_2^*(s) = \underset{a}{\text{argmax}}\ Q_2^*(s, a) = \underset{a}{\text{argmax}}\ (Q_1^*(s, a) - f(s))$$

$$\pi_1^*(s) = \underset{a}{\text{argmax}}\ Q_1^*(s,a)$$

Thus we can get that:

$$\boxed{\pi_1^*(s) = \pi_2^*(s)} \tag{2}$$

(b) (3 marks) Now assume that M_1 and M_2 has the same state transition probabilities but different reward functions. Let $R_1(s,a,s')$ and $R_2(s,a,s')$ give the expected immediate reward for the transition from s to s' under action a in M_1 and M_2, respectively. Given the optimal state-action value functions are related as given above, what is the relationship between the functions R_1 and R_2 ? That is, what is R_1 in terms of R_2 and f; OR R_2 in terms of R_1 and f.

Solution:

The Bellman optimality equation for the state-action value function can be written as follows:

$$Q^*(s,a) = \sum_{s',a} p(s',r|s,a)[r + \gamma \cdot \max_a Q^*(s',a)$$

However we know the relationship between $Q_1^*(s,a)$ and $Q_2^*(s,a)$. We can thus define the following:

$$Q_1^*(s,a) - Q_2^*(s,a) = f(s)$$

$$\Rightarrow \sum_{s',r} p(s',r|s,a)[R_1 - R_2 + \gamma \cdot \max_a Q_1^*(s',a) - \gamma \cdot \max_a Q_2^*(s',a)] = f(s)$$

$$\Rightarrow \sum_{s',r} p(s',r|s,a)[R_1 - R_2 + \gamma \cdot f(s')] = f(s)$$

$$\Rightarrow \sum_{s',r} p(s',r|s,a)[R_1 - R_2 + \gamma \cdot f(s')] = f(s)$$

Rearranging the above equation we can write:

$$\Rightarrow \boxed{R_1(s',a,s) = R_2(s',a,s) + \frac{f(s) - \gamma \cdot f(s') \cdot \sum_{s',r} p(s',r|s,a)}{\sum_{s',r} p(s',r|s,a)}} \tag{3}$$

The above equation gives us the relationships between functions R_1 and R_2.

4. (10 marks) [Jack's Car Rental] Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited $ 10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of $ 2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number n is $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night.

   (a) (4 marks) Formulate this as an MDP. What are the state and action sets? What is the reward function? Describe the transition probabilities (you can use a formula rather than a tabulation of them, but be as explicit as you can about the probabilities.) Give a definition of return and describe why it makes sense.

Solution: Any MDP with a deterministic reward can be formulated as:

$$MDP = \langle S, A, R, p \rangle$$

where, **S** denotes the state set, **A** denotes action set, **R** denotes rewards for every transition and **p** denotes the transition probabilities. Let us define the above for the given problem.

State Set: The state set S of the problem is defined as a 21×21 set with a general state denoted by (c_1, c_2) where $0 \leq c_1, c_2 \leq 20$ and the values of c_1 and c_2 respectively denote the number of cars at location 1 and location 2.

Action Set: The action set A can be visualised as movement of cars from location 1 to location 2. Thus we have $A = \{a | a \in \mathbb{Z}, -5 \leq a \leq +5\}$.

Rewards: The rewards for every action can be written using the number of cars rented at the particular location. Let x_i and y_i denote the number of cars rented or returned respectively at location i. Then reward for action a is:

$$R(a) = -2|a| + 10(x_1 + x_2)$$

Transition Probabilities: The transition between states at the end of every working day can be given by:

$$(c_1, c_2) \xrightarrow{a} (c_1'', c_2'') \longrightarrow (c_1', c_2')$$

Here c_i denotes the number of cars at location i at the end of the previous day. The action a performed on it refers to the nightly transfer of cars before business starts the next day. The number of cars at the start of the day is c_i''

at location i. We can see that c_1, c_2 and a deterministically give us c_1'' and c_2''. A few points to be noted about the values c_i, c_i' and c_i'' are:

- $c_1'' = min(c_1 - a, 20)$ and $c_2'' = min(c_2 + a, 20)$ - car movement
- $c_i' = c_i'' - x_i + y_i$ - based on car rental and returns
- $y_i = c_i' - c_i'' + x_i \geq 0$ - since car returns must be non-negative in number

Note that the above formulation of $y_i = c_i' - c_i'' + x_i$ offers a great deal of mathematical freedom while writing equations and solving. When y_i drops below zero we can set the probability of such cases to be zero. When we have an excess of cars the value of c_i' will remain 20 irrespective of this excess. Thus we look over a number of possibilities that may be lost with this upper bounding. To counter this, we let the excess at the end of the day at location i be denoted as $z_i = c_i'' - x_i + y_i - 20$.

Then the transition probability can be written as:

$$P(c_1', c_2'|c_1, c_2, a) = P(c_1', c_2'|c_1'', c_2'') = P(c_1'|c_1'') \cdot P(c_2'|c_2'')$$

For the i^th location the probability of a day starting with c_i'' cars and ending with c_i' cars can be calculated by summing over all possible combinations of cars being rented and returned:

$$P(c_i'|c_i'') = \sum_{x_i=0}^{c_i''} P(x_i|\lambda_i, c_i'') \cdot P(y_i = c_i' - c_i'' + x_i|\mu_i, c_i'')$$

Note that the values λ_i and μ_i denote the means of the poisson distributions corresponding to the car renting demand and return $[\lambda_1 = 3, \lambda_2 = 4, \mu_1 = 2, \mu_2 = 3]$. Note that $\forall y_i = c_i' - c_i'' + x_i < 0$, we must have $P(y_i|\mu_i, c_i'') = 0$.

Thus the complete formula for the transition probability can be written as:

Case 1: $c_1', c_2' < 20$

$$P(c_1'|c_1, c_2, a) = P(c_1'|c_1'') = \sum_{x_1=0}^{min(c_1-a,20)} \frac{3^{x_1}e^{-3}}{x_1!} \cdot \frac{3^{y_1}e^{-3}}{y_1!}$$

$$P(c_2'|c_1, c_2, a) = P(c_2'|c_2'') = \sum_{x_2=0}^{min(c_2+a,20)} \frac{4^{x_2}e^{-4}}{x_2!} \cdot \frac{2^{y_2}e^{-2}}{y_2!}$$

However this **equation holds only when number of cars is less than 20** ie. $c_1', c_2' < 20$. When we have an excess of cars the value of c_i' will remain 20

irrespective of this excess. Thus we look over a number of possibilities that may be lost with this upper bounding.

To counter this, we let the excess at the end of the day at location i be denoted as $z_i = c_i'' - x_i + ret_i - 20$. Then we may write the transition probabilities for this case as:

$$P(c_1' = 20|c_1, c_2, a) = \sum_{z_i=0}^{\infty} \sum_{x_1=0}^{min(c_1-a,20)} \frac{3^{x_1}e^{-3}}{x_1!} \cdot \frac{3^{y_1+z_1}e^{-3}}{(y_1+z_1)!}$$

$$P(c_2' = 20|c_1, c_2, a) = \sum_{z_i=0}^{\infty} \sum_{x_2=0}^{min(c_2+a,20)} \frac{4^{x_2}e^{-4}}{x_2!} \cdot \frac{2^{y_2+z_2}e^{-2}}{(y_2+z_2)!}$$

Note ret_i denotes the actual number of cars returned whereas y_i is simply a mathematical formaulation of the same which does not obey limitations of reality (ie. y_i may become negative).

Returns The formulation of the return in this case does not change significantly. We will have:

$$G_t = \sum_{k=1}^{\infty} \gamma^{k-1} \cdot R_{t+k}$$

This return equation can be used to control the business from running out of cars by giving a high negative reward for such states. Furthermore discounted return allows us to have a sort of look-ahead to avoid such states in the 'foreseeable' future.

(b) (3 marks) One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs $ 2, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of $ 4 must be incurred to use a second parking lot (independent of how many cars are kept there). These sorts of nonlinearities and arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. Can you think of a way to incrementally change your MDP formulation above to account for these changes?

Solution: There are two non-linearities introduced by the above modification of the problem.

Non-Linearity 1

The employee who offers to transport one car for free from location 1 to location

2 can be represented by using the following adjustment. Note that the use of this employee's proactiveness need only be made in case there is a transfer of cars from location 1 to 2 overnight but not from location 2 to location 1. The modified reward can be written as:

$$R(a) = -2 \cdot min(|a|, |a-1|) + 10 \cdot (x_1 + x_2)$$

Non-Linearity 2

The extra cost incurred by the requirement of a second parking lot at the end of the day (including the overnight shifting of cars) means that there must not exceed 10 cars at a location irrespective of the overnight movement of cars. Thus if at the end of the day we have c_i cars at location i and the action selected for the day is a we have:

$$c_1'' = c_1 - a > 10 \Rightarrow R(a) \leftarrow R(a) - 4$$
$$c_2'' = c_2 + a > 10 \Rightarrow R(a) \leftarrow R(a) - 4$$

(c) (3 marks) Describe how the task of Jack's Car Rental could be reformulated in terms of *afterstates*. Why, in terms of this specific task, would such a reformulation be likely to speed convergence? *(Hint:- Refer page 136-137 in RL book 2nd edition. You can also refer to the video at https://www.youtube.com/watch?v=w3wGvwi336I)*

Solution: In the above problem we can treat the set of values (c_1'', c_2'') as a set of afterstates. Different states which move to the same afterstate at the end of the day after the action a is applied on them can now be treated equivalently by mapping such (c_1, c_2, a) tuples to the pair (c_1'', c_2''). This allows us to greatly improve the convergence since we no longer need to worry about the extra dimensionality introduced by the action a ie. the mapping is now inherently simpler.

5. (8 marks) [Organ Playing] You receive the following letter:
Dear Friend, Some time ago, I bought this old house, but found it to be haunted by ghostly sardonic laughter. As a result it is hardly habitable. There is hope, however, for by actual testing I have found that this haunting is subject to certain laws, obscure but infallible, and that the laughter can be affected by my playing the organ or burning incense. In each minute, the laughter occurs or not, it shows no degree. What it will do during the ensuing minute depends, in the following exact way, on what has been happening during the preceding minute: Whenever there is laughter, it will continue in the succeeding minute unless I play the organ, in which case it will stop. But continuing to play the organ does not keep the house quiet. I notice, however, that whenever I burn incense when the house is quiet and do not play the organ it remains quiet for the

next minute. At this minute of writing, the laughter is going on. Please tell me what manipulations of incense and organ I should make to get that house quiet, and to keep it so.
Sincerely,
At Wits End

(a) (4 marks) Formulate this problem as an MDP (for the sake of uniformity, formulate it as a continuing discounted problem, with $\gamma = 0.9$. Let the reward be +1 on any transition into the silent state, and -1 on any transition into the laughing state.) Explicitly give the state set, action sets, state transition, and reward function.

Solution: Any MDP with a deterministic reward can be formulated as:

$$MDP = \langle S, A, R, p \rangle$$

where, **S** denotes the state set, **A** denotes action set, **R** denotes rewards for every transition and **p** denotes the transition probabilities. Let us define the above for the given problem.

State Set: The state set S of the problem is defined as:

$$S = L, Q$$

where L denotes ghostly laughter and Q denotes quietude.

Action Set: The action set A comprises of all possible combinations of playinng the organ (O) and lighting incense(I):

$$A = \phi, O, I, IO$$

where ϕ denotes no action and IO denotes that both are taking place.

Rewards: The rewards for every action that results in quietude is +1 and for every action with the ghostly laughter is -1.

Transition Probabilities: The transition between states can be formulated in the following state transition table:

State	Action	Next State	Reward	Trans. Probability
Q	ϕ	L	-1	1.00
Q	I	Q	+1	1.00
Q	O	L	-1	1.00
Q	IO	L	-1	1.00
L	ϕ	L	-1	1.00
L	I	L	-1	1.00
L	O	Q	+1	1.00
L	IO	Q	+1	1.00

We can see that the effect of an action is deterministic and so is the reward.

(b) (2 marks) Starting with simple policy of **always** burning incense, and not playing organ, perform a couple of policy iterations.

Solution: Consider the above policy. Then we have $\pi_0(L) = I$ with reward -1 and $\pi_0(Q) = I$ with reward $+1$. Further before we carry out policy iteration we can set the values of states as: $V(Q) = V(L) = 0$. For the purposes of policy evaluation we let $\theta = 0.5$.

Now we carry out the iterations:

ITERATION 1

Policy Evaluation We know that the value function of the policy can be evaluated as:

$$V_\pi(s) = \sum_{s',r} p(s', r|s, \pi(s)) \cdot [r + \gamma V_\pi(s')] = (r + 0.9V(s'))$$

We carry this out until the value of $max(\Delta V(L), \Delta V(Q)) = \Delta V$ becomes less than θ.

$V(L) = -1 + 0.9(0) = -1.0$	$V(Q) = 1 + 0.9(0) = 1.0$	$\Delta V = 1$
$V(L) = -1 + 0.9(-1) = -1.9$	$V(Q) = 1 + 0.9(1) = 1.9$	$\Delta V = 0.9$
$V(L) = -1 + 0.9(-1.9) = -2.71$	$V(Q) = 1 + 0.9(1.9) = 2.71$	$\Delta V = 0.81$
$V(L) = -1 + 0.9(-2.71) = -3.44$	$V(Q) = 1 + 0.9(2.71) = 3.44$	$\Delta V = 0.73$
$V(L) = -1 + 0.9(-3.44) = -4.10$	$V(Q) = 1 + 0.9(3.44) = 4.10$	$\Delta V = 0.66$
$V(L) = -1 + 0.9(-4.10) = -4.69$	$V(Q) = 1 + 0.9(4.10) = 4.69$	$\Delta V = 0.59$
$V(L) = -1 + 0.9(-4.69) = -5.22$	$V(Q) = 1 + 0.9(4.69) = 5.22$	$\Delta V = 0.53$
$V(L) = -1 + 0.9(-5.22) = -5.70$	$V(Q) = 1 + 0.9(5.22) = 5.70$	$\Delta V = 0.48$

Policy Improvement We know that the policy improvement is done at each step as:

$$pi(s) \longleftarrow \underset{a}{argmax} \sum_{s'} p(s'|s, a) \cdot (E[r|s, a, s'] + \gamma \cdot V(s'))$$

Then we can get:

$$pi(L) \in \{O, IO\}$$
$$pi(Q) = I$$

ITERATION 2

Policy Evaluation We carry out evaluation till the value of $max(\Delta V(L), \Delta V(Q)) = \Delta V$ becomes less than $\theta = 0.5$.

$$V(L) = 1 + 0.9(5.70) = 6.13 \quad V(Q) = 1 + 0.9(5.70) = 6.13 \quad \Delta V = 0.43$$

We can see that the policy evaluation ends in the first step itself.

Policy Improvement

Now we attempt to improve the policy.

$$pi(s) \longleftarrow \underset{a}{argmax} \sum_{s'} p(s'|s,a) \cdot (E[r|s,a,s'] + \gamma \cdot V(s'))$$

Then we can get:

$$pi(L) \in \{O, IO\}$$

$$pi(Q) = I$$

The policy is unchanged. Thus the above policy is a **STABLE POLICY**.

(c) (2 marks) Finally, what is your advice to "At Wits End"?

Solution: My advice to my friend 'AWE' would be to do the following:

- When there is laughter, **play organ to silence it**.
- This can be accompanied with burning of incense.
- Once silence is established **burn incense to maintain quietude**
- Do not play organ when it is quiet

6. (4 marks) [Stochastic Gridworld] An ϵ-greedy version of a policy means that with probability 1-ϵ we follow the policy action and for the rest we uniformly pick an action. Design a stochastic gridworld where a deterministic policy will produce the same trajectories as a ϵ-greedy policy in a deterministic gridworld. In other words, for every trajectory under the same policy, the probability of seeing it in each of the worlds is the same. By the same policy I mean that in the stochastic gridworld, you have a deterministic policy and in the deterministic gridworld, you use the same policy, except for ϵ fraction of the actions, which you choose uniformly randomly.

(a) (2 marks) Give the complete specification of the world.

Solution: In order to specify a stochastic gridworld which mimics an ϵ-greedy policy we first set up a sense of direction. Let the direction of chosen action be designated **NORTH**. Then in the stochastic gridworld:

- Agent moves in the direction of chosen action with probability $1 - \frac{3\epsilon}{4}$

- Agent moves in **WEST** direction with probability $\frac{\epsilon}{4}$
- Agent moves in **EAST** direction with probability $\frac{\epsilon}{4}$
- Agent moves in **SOUTH** direction with probability $\frac{\epsilon}{4}$

(b) (2 marks) Will SARSA on the two worlds converge to the same policy? Justify.

Solution: YES
SARSA updates take place on a current state depending on the action.The underlying MDPs and the Bellman equations are same for the deterministic and stochastic gridworlds as defined above. Furthermore in terms of transition probabilities and expected rewards, the expectations of the actual movements of the agent will be identical and thus expected SARSA updates will be the same. Thus SARSA on both worlds will estimate the same action-value functions and will eventually converge to the same policy at the end of

7. (5 marks) [Contextual Bandits] Consider the standard multi class classification task (Here, the goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs). Can we formulate this as contextual bandit problem (Multi armed Bandits with side information) instead of standard supervised learning setting? What are the pros/cons over the supervised learning method. Justify your answer. Also describe the complete Contextual Bandit formulation.

Solution: YES

It is easy to re-formulate the multi-class classification problem as a contextual bandit problem. In order to do so we have to draw a parallelism between the classification of a data point into one of many classes as done in classification to the prediction of the optimal arm in the contextual bandit problem. The following is a description of the complete contextual bandit formulation:

- **Features**: The features of the image (for an image classification problem) or the data point provide the context to the contextual bandit. The data maybe used directly to do so or in the form of a feature vector obtained from the raw data.
- **Classes**: The classes for the above classification problem can be represented as the arms of the bandit. The problem now is to find out which is the optimal arm for each context.

- **Training**: For each context (data point) the optimal arm corresponds to the correct class and has a reward of $+\lambda$ and other arms have a reward of 0. In each iteration we try to maximise the reward corresponding to each context.

PROS

The major pros for this method is that the training happens in an online fashion improving itself after each iteration. Furthermore some papers claim that the performance of such models in the context of error-free classification is better than simple MLP based classifiers.

CONS

The major con for this method is that the amount of training data remains the same but for each training instance (data point) multiple iterations are required to learn the best arm (class) for each context. The model needs to observe many context vectors before it can achieve a satisfactory accuracy, the initial performance of the model will be very poor. The computational resources required by this method are also much more than that of standard supervised learning methods, a larger number of epochs is usually needed for training.

8. (5 marks) [TD, MC, PG] Suppose that the system that you are trying to learn about (estimation or control) is not perfectly Markov. Comment on the suitability of using different solution approaches for such a task, namely, Temporal Difference learning, Monte Carlo methods, and Policy Gradient algorithms. Explicitly state any assumptions that you are making.

Solution:

Temporal Difference Learning; **NO**

Temporal difference learning would not work for non-markovian systems. For these systems transition probabilities and expected rewards depend on the entire history of the problem and not just on the current state. Thus the TD updates made to the value function of a state-action pair for various sequences leading to that state, will each point to a different target every time. Thus the policies learnt as well as the value function estimated need not be correct for such systems.

Monte Carlo Methods: **YES**

Monte Carlo methods (first-visit or every-visit) essentially do not make use of the Markov assumption. Thus they will work on non-Markovian problems as well. As long as there exist well-defined rewards and the tasks are performed episodically, Monte Carlo methods would work by sampling sequences of states, actions and rewards.

Policy Gradient Algorithms: **NO** In the proof of the policy gradient theorem, we invoke the Markov assumption in order to prove it. Thus for a non-markovian system, we cannot use policy gradient algorithms, since it is crucial for the the states to contain all the information available to make a decision.

9. (5 marks) [PG] Recent advances in computational learning theory, have led to the development of very powerful classification engines. One way to take advantage of these classifiers is to turn the reinforcement learning problem into a classification problem. Here the policy is treated as a labeling on the states and a suitable classifier is trained to learn the labels from a few samples. Once the policy is adequately represented, it can be then used in a policy evaluation stage. Can this method be considered a policy gradient method? Justify your answer. Describe a complete method that generates appropriate targets for the classifier.

Solution:

Ref: Reinforcement Learning as Classification, Lagoudakis Parr, ICML 2003

The algorithm detailed in the above paper allows us to convert the reinforcement learning problem into a classification problem. The following steps are followed:

- **Q-Value Estimation**

  We use a fixed number of samples to generate approximate values for the state-action value function by carrying out several rollouts.

- **Positive Training Examples**

  We generate positive examples for $(s, a^*)^+$ if the action a^* is statistically significant actions for a particular state, ie. actions for which the state-action value function is significantly bigger than for every other action $a \in A$.

- **Negative Training Examples**

  We generate positive examples for $(s, a)^-$ if the action a is statistically insignificant actions for a particular state, ie. actions for which the state-action value function is significantly lower than for the action a^*.

The examples are fed into a classification model for training. Each state-action pair will be represented by input features and the classifier can classify correctly as positive or negative.

YES - This method can be considered as a policy gradient method. We use parametrised method for learning and the parameters are the ones used in the classifier for solving it.