CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55
Name: Royyuru Sai Prasanna Gangadhar **Roll number:** ME19B190

- 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: $\mu(s) = Q^*(a_i)$
   $E(\frac{R_i}{a_i}) = \sum_a (E[\frac{R}{a}]).P(\frac{a}{a_i}) = (1-\epsilon).Q^*(a_i) + \sum_{j=1}^{N} \frac{\epsilon}{N}.Q^*(a_j)$

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: Because there is a delay of τ units, there is no reward that the agent receives till the first τ time steps. Therefore return can be written as

   $$G_n = \Sigma_{t=n+\tau}^{\infty} (\gamma)^t . r_t$$

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

Solution: TD update(δ_{TD})(SARSA) $= r_{t+1} + \gamma Q(s_{t+1}, a_{t+1-\tau} - Q(s_t, a_{t-\tau})$
TD update(δ_{TD})(Q learning) $= r_{t+1} + \gamma(max_a Q(s_{t+1}, a)) - Q(s_t, a_{t-\tau})$
$Q(s_t, a_{t-\tau}) \leftarrow Q(s_t, a_{t-\tau}) + \alpha *$ **(TD update)**

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 $f : S \rightarrow R$:

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

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: $\pi_2^*(a|s) = argmax_a Q_2^*(s, a)$
$= argmax_a(Q_1^*(s, a) - f(s))$, f(s) is independent of actions "a", therefore
$\pi_2^*(a|s) = argmax_a Q_1^*(s, a) = \pi_1^*(a|s)$
Thus, M_1 and M_2 have same optimal policies

(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: Given same state transition probabilities. From Bellman optimality equation we get,

$$Q^*(s, a) = \Sigma_{s'} P(s^{'}, r|s, a)[r + \gamma.max_a Q^*(s^{'}, a)]$$

$$Q^*(s, a) = \Sigma_{s^{'},r} P(s^{'}, r|s, a).r + \gamma \Sigma_{s'} P(s^{'}, r|s, a) max_a Q^*(s^{'}, a)$$

$$Q^*(s, a) = \Sigma_{s'} P(s^{'}|s, a)[R(s, a, s^{'}) + \gamma max_a Q^*(s^{'}, a)]$$

Therefore now, substituting for Q_1 and Q_2 we get,

$$Q_1^*(s, a) = \Sigma_{s'} P(s^{'}|s, a)[R_1(s, a, s^{'}) + \gamma max_a Q_1^*(s^{'}, a)]$$

$$Q_2^*(s,a) = \Sigma_{s'} P(s^{'}|s,a)[R_2(s,a,s^{'}) + \gamma max_a Q_2^*(s^{'},a)]$$

Given that both the finite MDP's have the same optimal policies and assuming gamma = 1(as it is finite MDP), subtracting the above two equations we get

$$f(s) = \Sigma_s^{'} P(s^{'}|s,a).[R_1(s,a,s^{'}) - R_2(s,a,s^{'}) + f(s^{'})]$$

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 1re 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: Let the two locations be 1 and 2.

States = pair indicating the number of cars at location 1 and location 2 at the end of each day- $[N_1, N_2]$. Given maximum number of cars at each location is 20, Thus $N_1, N_2 \in [0, 20]$. Thus, there are a total of 441 (21 * 21 states).

Actions: Number of cars moved between locations A and B overnight. Thus, action space $A(S)$ is

$$A(S) \in [-5, 5]$$

negative values of action indicate cars are moved from location 2 to 1 and positive values indicate cars are moved from 1 to 2.

State transition probabilities: Let r_1, r_2 be the rental requests at locations 1 and 2 and R_1, R_2 be the returns at locations 1 and 2. From question r_1, r_2 follows poisson(x, $\lambda = 3$), poisson(x,$\lambda = 4$) and R_1, R_2 follows poisson(x,$\lambda = 3$), poisson(x,$\lambda = 2$). From now poisson is abbreviated as Po. Thus, the transition

probability is given as:

$$s \xrightarrow{a} s^{'}$$

$$s^{'} = s + [-action, action] - [r_1, r_2] + [R_1, R_2]$$

$$P(s^{'}|s, a) = Po(r_1, \lambda = 3).Po(r_2, \lambda = 4).Po(R_1, \lambda = 3).Po(R_2, \lambda = 2)$$

Reward function: $(r_1 + r_2)$*10 is the reward we get for renting the cars available at location 1 and 2, and it costs \$ 2 per car moved. Thus, a negative reward of $2.a$ (2 . number of cars moved) is incurred. Thus, the net instantaneous reward is $((r_1 + r_2).10 - 2.a)$. Thus, the expected reward for a state action pair $r(s, a)$ is given by:

$$r(s, a) = \Sigma_{r_1}\Sigma_{r_2}\Sigma_{R_1}\Sigma_{R_2}P(s^{'}|s, a).((r_1 + r_2).10 - 2.a)$$

Since, the Poisson random variable can take any real positive interger value, it is better to truncate the above summation to an upper limit to speed up convergence. As the probability of very large requests is highly unlikely, this truncation does not affect our analysis greatly. The rewards are only given when the agent reaches the next state but not at intermediate locations. The total reward that the owner receives at end of each day can be defined as the sum of reward by renting out cars in both locations minus cost incurred in transportation from location 1 to 2. Thus $r_t = (r_1 + r_2) * 10 - 2.a$, where r_1, r_2 are the number of cars that were rented on that particular day and a is the number of cars moved from location 1 to 2. Then the cumulative reward with time is given by:

$$G_t = \Sigma_{t=0}^{\infty}\gamma^t r_t, \;\; r_t = (r_1 + r_2) * 10 - 2.a$$

(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: Initialize as policy(deterministic) $\pi(s)$.
Initialize the returns for all states arbitrarily (say zeros).

1. Loop:
perform action a,in the "currentstate" following $\pi(s)$. Maximum number of cars at each location cannot exceed 20. Therefore,

$$cars_1 = min(currcars_1 - action, 20)$$

$$cars_2 = min(currcars_2 + action, 20)$$

Since an employee is willing to shuttle a car from 1 to 2 for free, reward we get for shuttling between the two locations is
r(currentstate,a) += -2*[abs(action) if action $\leq$ 0 else abs(action)-1]
Another constraint is on number of cars parked should not exceed 10 overnight. Therefore, reward we incur if the cars exceed 10 at any location is:

$$cars_1 > 10 \rightarrow r(currentstate, a) += -4$$

$$cars_2 > 10 \rightarrow r(currentstate, a) += -4$$

2. Loop for all possible r_1, r_2, R_1, R_2:

$$rented_1 = min(r_1, cars_1); rented_2 = min(r_2, cars_2)$$

$$r(currentstate, a) += P(s^{'}|s, a).((rented_1 + rented_1) * 10) \quad (1)$$

if we want to perform policy iteration or policy evaluation, r(s,a) is replaced by $V_\pi(S)$ and the equation 1 is replaced by

$$nextstate = [cars_1, cars_2] + [-action, action] - [r_1, r_2] + [R_1, R_2]$$

$$V_\pi(currentstate) += P(s^{'}|s, a).((rented_1 + rented_1) * 10 + \gamma.V_\pi(nextstate)) \quad (2)$$

$$currentstate = nextstate$$

, repeat from 1.

(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: Let us define the afterstate S_a as the state that results after Jack performs the action a in current state . Illustration of the transition diagram is given below:

$$S \xrightarrow{a} S_a \xrightarrow{P(S^{'}|S_a)} S^{'}$$

As it is evident S_a is also a subset of the state space for this example. Now thus the update rule to the value of afterstates under a deterministic policy π can be written as:

Update to after states: (In-order to be concise only the update rule is given not the entire iteration algorithm)

$$V(S_a) += \Sigma_{r_1}\Sigma_{r_2}\Sigma_{R_1}\Sigma_{R_2}P(s^{'}|S_a).[((r_1 + r_2).10 - 2.a) + \gamma * V(S^{'})] \quad (3)$$

The above equation is looped over all states and possible actions, until convergence. The afterstates effectively learn the action value function of a state action pair. We then select the action in a state that results in best afterstate i.e, best value of afterstate:

Selecting actions based on afterstates:

$$\pi(S) \leftarrow argmax_a \; V(S_a) \quad (4)$$

This reformulation is likely to speed convergence because, there are many state action pairs that result in the same afterstate. Thus, all these state action values are learnt once instead of learning them separately. Another advantage in this formulation is that, afterstates are also a subset of state space for this example. Thus the value of states under a deterministic policy receives more updates in comparison with standard policy iteration, thus it leads to faster convergence.

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:

States : There are two states laughter and no laughter/silent state

actions : Four actions are possible in each state namely (play organ and burn incense), (play organ and don't burn incense), (don't play organ and burn incense) and (don't play organ and don't burn incense). Thesee are represented as $O \cap I$, $O \cap I^-$, $O^- \cap I$ and $O^- \cap I^-$ respectively. The following table represents the state set, action set, state transition and reward function

State (s)	Action(a)	Next state(s$'$)	r(s,a,s$'$)
laughter	$O \cap I$	no laughter	+1
laughter	$O \cap I^-$	no laughter	+1
laughter	$O^- \cap I$	laughter	-1
laughter	$O^- \cap I^-$	laughter	-1
no laughter	$O \cap I$	laughter	-1
no laughter	$O \cap I^-$	laughter	-1
no laughter	$O^- \cap I$	no laughter	+1
no laughter	$O^- \cap I^-$	laughter	-1

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

Solution: Policy iteration: Intialize states V(l), V(nl) = 0 and initialize a policy $\pi(s) = O^- \cap I$, where l,nl represent laughter and silent (no laughter) states respectively.

Policy evaluation:

Loop for each state and loop until convergence:

$V(S) \leftarrow \Sigma_{s'} \Sigma_r P(s', r|s, a)(r + \gamma V(s'))$

we get

$$V(l) = -1 + -1.(0.9) + -1.(0.9^2) + -1.(0.9^3) + ... = -10$$

$$V(nl) = 1 + 1.(0.9) + 1.(0.9^2) + 1.(0.9^3) + ... = 10$$

Policy improvement:

$$policystable \leftarrow True$$

$$a = O^- \cap I$$

$$\pi(l) = O \cap I$$

$$policystable \leftarrow false$$

similarly,

$$\pi(l) = O^- \cap I$$

When the policy iteration is performed for the second time, we get:
V(l) = 1.9
V(nl) = 10

Policy improvement:

$$policystable \leftarrow True$$

$$a = O^- \cap I$$

$$\pi(l) = O \cap I$$

similarly,

$$\pi(l) = O^- \cap I$$

$$policystable \leftarrow True$$

Thus the policy has converged to the optimal policy.

Therefore V(l) = 1.9 and best action to take during a laughter state is to play the organ (irrespective of whether we burn the incense or not) and the best action to take in no laughter state is to burn the incense and not play organ.

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

Solution:

Play the organ when you hear laughter.

Burn the incense and do not play organ when there is no laughter i.e when the house 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 this given grid world $|A| = |S| = 4$.
Policy in Deterministic World:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|}, & \text{if a} = a_\pi \\ \frac{\epsilon}{|A|}, & \text{otherwise} \end{cases}$$

State transitions in deterministic world:

$$P(s^{'}|s, a) = 1$$

Policy in stochastic World:

$$\pi(a|s) = \begin{cases} 1, & \text{if a} = a_\pi \\ 0, & \text{otherwise} \end{cases}$$

State transitions in deterministic world:

$$\Sigma_{s'} P(s^{'}|s, a) = 1$$

$$\text{Trajectory in deterministic world}(t_d) = s_o \xrightarrow{a_o} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} .....$$

$$\text{Trajectory in stochastic world}(t_s) = s_o \xrightarrow{a_o} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} .....$$

Probability of seeing the trajectory in deterministic world

$$P(t_d) = \Pi_{i=0}^{end} \pi(a_i|s_i) \tag{5}$$

Probability of seeing the trajectory in stochastic world

$$P(t_s) = \Pi_{i=0}^{end} P(s_{i+1}|s_i, a_{s_i}) \tag{6}$$

For the probabilities of seeing the same trajectories to be equal, we need to equate the two equations 5 and 6 and we get the state transition probabilities of the stochastic world are:

$$P(S_{t+1}|s_t, a_{\pi(s_t)}) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|}, & \text{if } S_{t+1} = S_{a_{\pi(s_t)}} \text{(state in the direction of } a_{\pi(s_t)}) \\ \frac{\epsilon}{|A|}, & \text{otherwise,i.e for remaining 3 states} \end{cases}$$

Other assumptions involved are: both the worlds have same state action pairs. The probability distribution of the initial states in both of the worlds is the same. The expected reward for transition in both the worlds is the same.

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

Solution: Yes, SARSA on the two worlds converge to the same policy Let us rephrase the question to make our analysis simple. Will both the worlds have same optimal policy? (as SARSA will converge to the optimal policy)

$$\pi^*_{deterministic}(a|s) = argmax_a \Sigma_{s',r} P(s', r|s, a)[r + \gamma V(s')]$$

given that it is a deterministic grid world the above expression reduces to,

$$\pi^*_{deterministic}(a|s) = argmax_a[r(s \rightarrow s'_{t+1}) + \gamma V_*(s'_{t+1})]$$

Now consider the optimal policy in the stochastic world:

$$\pi^*_{stochastic}(a|s) = argmax_a \Sigma_{s',r} P(s', r|s, a)[r + \gamma V(s')]$$

$$= argmax_a(1-\epsilon+\frac{\epsilon}{|A|}[r(s \rightarrow s'_{t+1})+\gamma V_*(s'_{t+1})]+\Sigma_{s_{t+1} \neq s'_{t+1}} \frac{\epsilon}{|A|}[r(s \rightarrow s_{t+1})+\gamma V_*(s_{t+1})]$$

$$= argmax_a(1-\epsilon)[r(s \rightarrow s'_{t+1}) + \gamma V_*(s'_{t+1})] + \Sigma_{s_{t+1}} \frac{\epsilon}{|A|}[r(s \rightarrow s_{t+1}) + \gamma V_*(s_{t+1})]$$

In the above expression, right hand side of addition operator is independent of action "a" as it is the summation over all the next states which is same for all actions "a". Thus, the optimal policy reduces to

$$\pi^*_{stochastic}(a|s) = argmax_a(1-\epsilon)[r(s \rightarrow s'_{t+1}) + \gamma V_*(s'_{t+1})]$$

$$= argmax_a[r(s \rightarrow s'_{t+1}) + \gamma V_*(s'_{t+1})] = \pi^*_{deterministic}(a|s)$$

Therefore, SARSA in both the worlds converge to the same policy.

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, we can formulate this as a contextual bandit problem. Let us say we have k labels, we can use k bandits as representative for k classes. Let us use the reward that each bandit gets as the cross entropy loss of classifying the data. Thus, each bandit is initialized with a policy of random parameters over the possible actions. Here,actions for a bandit "i" is to output the likelihood that it thinks the data point

belongs to label i or not. Each bandit outputs the distribution of the probability that it thinks it belongs to class 'i'. We then calculate the reward that each bandit received using the cross entropy loss. Then the parameters of bandits are updated using policy gradient theorem. For a new data point, we choose the arm that gives the highest reward based on predicted probability distribution over learned parameters. Another approach could be, a random arm is selected for a given data point, the reward(cross entropy loss or some equivalent metric) is observed for pulling the random arm. Given a context(i.e a data point), we can use algorithms like Lin-UCB to select the arm that serves as a best estimator out of k arms. We can then classify the data point based on the arm selected. The biggest advantage of contextual bandits it allows us to predict the best option for each context setting, not just the best option overall. Another advantage of contextual bandits is that they perform comparatively better where the output label is not known. Lin UCB gives better performance with lesser training data. Some successful implementation of contextual bandits can also be found in real-world problems often involving underlying processes that are dynamically evolving over time. One of the disadvantages of contextual bandits is that they might attach low probabilities to the labels as it might not explore the entire space of outcomes. Hyperparameter tuning needs to be performed to find the optimal explore-exploit strategy. If the rewards received are of very high variance, it is difficult to implement contextual bandits, as it makes difficult when to stop the training of model.

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: Monte Carlo Methods: The Monte Carlo methods does not need a perfect Markov process to estimate the value functions as it returns the estimated value of complete return over multiple experiments. As the states are sampled from the environment, we will be able to find the true state value even if the environment dynamics is not completely Markov.

$$V(s) = E[G_t = \Sigma\gamma^t r_t | S_t = s]$$

Temporal difference learning: The one step TD target is given by

$$G_{t:t+1} = R_{t+1} + \gamma V(s_{t+1})$$

$$V(s) = E[G_{t:t+1} | S_t = s]$$

The above formulation assumes that the state s_{t+1} is sampled in the trajectory following the Markov decision process i.e immediate rewards and state transitions are s_{t+1} are determined by $P(r, s_{t+1}|s, a)$ and $\pi(a|s)$. Thus, the TD target is correct when and only when the process follows Markov dynamics.

Policy Gradient: The answer for this question might depend on our implementation. If we are using Monte Carlo policy gradient theorem, then our policy gradient will be able to handle handle non Markov processes. But if we are using actor critic methods with TD returns then, we again implicitly assume that our process follows Markov dynamics and thus might not be able to perform well over non markov dynamics.

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: In policy gradient methods, we calculate the return following a given parameterized policy. The policy is updated by using the gradient with respect to the current policy parameters. But when we consider the task of formulating the reinforcement learning problem as a classification problem, we try to predict the label of each state in a supervised learning manner from samples given. Thus formulating the RL problems as a classification problem is different from policy gradient methods. The loss in classifier model is used to update the parameters of classifier model, but the return obtained is used to update the value of parameters of policy in policy gradient theorem. Thus, this method cannot be considered a policy gradient approach.Although both of them seem updating the parameters,both the approaches are fundamentally different.

To generate targets we can use set of states labeled with correct actions. This labeled data set should be used to train a classifier using supervised learning. This classifier then can be used to generate the appropriate targets for the classifier. Thus, the targets for the classifier can be provided by the labels provided by the trained classifier.

1. Given limited data: $(x_1, y_1), (x_2, y_2), ...$, the x_i can be the states and y_i can be the corresponding label of that particular state.

2. Train a classifier using the given data using supervised learning: example a neural network

3. Now input the states that we need label for, to the trained neural network

4. Now the output of the neural network can act as a target for the classifier.