Lecture - 2

Recap from last lecture:

1) Introduction about RL framework.
2) Description of Markov Decision Process with example of cartpole balance task
3) Dynamic Programming based algorithm for finite horizon
4) Value iteration and policy iteration algorithm for infinite horizon
5) Stationary vs non stationary policy
6) Model based v/s model free algorithm

Topics to be covered today:

Value Iteration type scheme:

1) Q learning
   - Tabular
   - Function approximation

2) SARSA
   - Tabular
   - Function approximate

Policy evaluation schemes:

3) Monte Carlo Policy evaluation

4) TD(0) Policy evaluation

5) Gradient TD(0) Policy evaluation

Q-Learning (Tabular)

- State and action space is finite
- We do not assume access to reward function and transition probability. Hence Q-learning is model free
- Infinite horizon tasks are considered.
- Discounted reward setting is assumed.
- Policy is stochastic and stationary.

Let us define Q-value function for stochastic policy π:

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = s, A_0 = a\right]$$

$$Q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\left(R(s,a,s') + \gamma\, E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = s', A_0 = a\right]\right)$$

$$Q^{\pi}(s,a) = \sum P(s'|s,a)\left(R(s,a,s') + \gamma \sum \pi(a'|s') E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_1 = s', A_1 = a' \right] \right)$$

$$\Rightarrow \boxed{Q^{\pi}(s,a) = \sum P(s'|s,a)\left(R(s,a,s') + \gamma \sum \pi(a'|s') Q^{\pi}(s',a') \right)}$$

$\Downarrow$

- This is the Bellman equation satisfied by the Q-value function

- Based on this Bellman equation a value iteration based scheme is devised to obtain the optimal Q-value function and hence optimal policy. This scheme is called Q-learning

- Q learning uses ε-greedy policy:

$$\pi(a|s) = \begin{cases} 1 - \varepsilon + \varepsilon/|A| & , a = \underset{a}{\operatorname{argmax}}\, Q(s,a) \\ \varepsilon/|A| & , \text{otherwise} \end{cases}$$

Q-learning algorithm:

1) Initialise value of $Q(s,a)$ for all states and all actions.

2) Initialize S

- Till the episode terminates or max timestep limit is reached perform following
- Take action A according to ε-greedy policy and observe reward R and next state S'
- Update Q-value:

$$Q(S,A) = Q(S,A) + \alpha \left(R(S,A,S') + \gamma \max_a Q(S',a) - Q(S,A) \right)$$

- $S \leftarrow S'$

3) Repeat step 2 untill convergence

Explanation:

1) $y = E[X]$

$x_0, x_1, x_2, x_3, x_4 \text{ ---- }$

$y_0 = x_0$

$y_1 = \frac{x_0 + x_1}{2}$

$y_{n-1} = \frac{x_0 + x_1 + \text{-----} x_{n-1}}{n}$

$y_n = y_{n-1}\left(\frac{n}{n+1}\right) + x_n\left(\frac{1}{n}\right)$

$\alpha = 1/n$

$y_n = y_{n-1}(1-\alpha) + x_n \alpha$

$Q(s,a) = Q(s,a)(1-\alpha) + \alpha\left(R(s,a,s') + \gamma \max_{a'} Q(s',a')\right)$

- We are trying to find $y = E[x]$. Hence we use scheme:

$$y_n = y_{n-1}(1-\alpha) + x_n \alpha$$

- When we use the scheme:

$$Q_n(s,a) = Q_{n-1}(s,a)(1-\alpha) + \alpha\left(R(s,a,s') + \gamma \max_{a'} Q(s',a')\right)$$

We are trying to find $Q(s,a)$ such that

$$Q(s,a) = E\left[R(s,a,s') + \gamma \max_{a'} Q(s',a')\right]$$

Points to notice:

1) We are using ε-greedy policy for collecting data s, a, s'
2) However we are find Q-value of greedy policy.

Greedy policy is clear from the Q-value update equation:

$$Q(s,a) \leftarrow Q(s,a)(1-\alpha) + \alpha\left(R(s,a,s') + \gamma \max_{a'} Q(a',s')\right)$$

- Q-learning is off-policy algorithm because we use ϵ-greedy policy to generate data but we find Q-value function for greedy policy.

- Difference between off-policy and on-policy algorithm will be more clear when we discuss SARSA algorithm.

Q-learning with Function Approximation

Q) Let no. of states = 10^8 and no. of actions = 10. What is the memory requirement of tabular Q-learning?

→ No of enteries to store = $10^8 \times 10$

$= 10^9$

Let each entry takes 4 bytes of space.

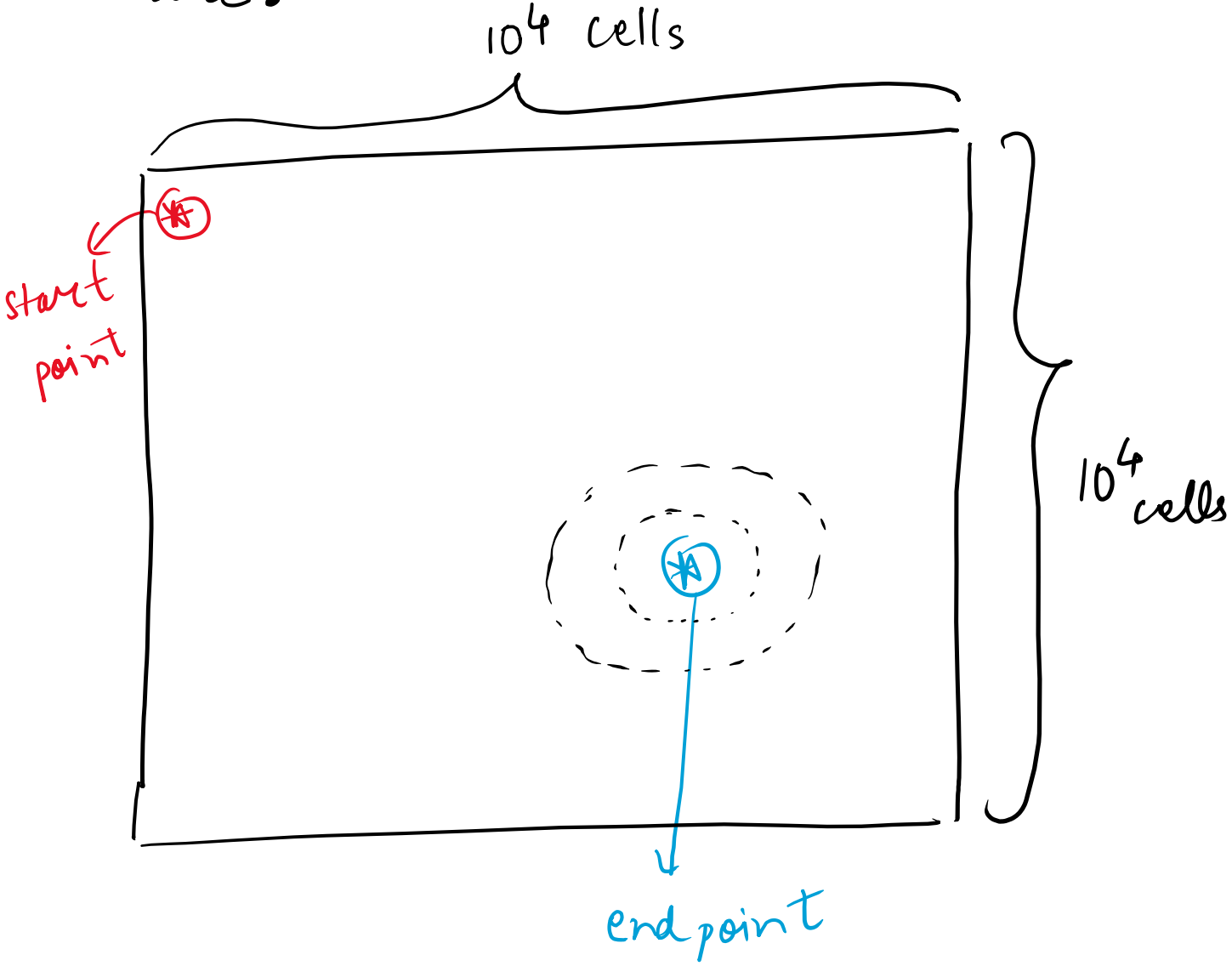
Total memory requirement

$= 4 \times 10^9$ bytes $\approx$ 4 GB

- This implies Q-learning algorithm will take up 4 GB of RAM.
- Further learning Q-values for 10^9 (s,a) pairs means taking 10^9 environment steps.

- Using tabular Q-learning is not economical for problems with large state space both memory wise and time wise.

- Let us consider an example of MDP where nearby states will have almost similar value.



- Let us say the reward for ending up at end state is +100 and otherwise it is 0.
- The states which are close to end state will have value $V(s)$ close to 100.
- Further states close to each other under a certain policy will have same value $V(s)$.
- Exploiting this observation we can write the value function and Q-value function as $\hat{v}(s) = \phi(s)^T w$ and $\hat{Q}(s,a) = \phi(s,a)^T w$.
- Now we don't need to learn about 10^8 state values, rather finding the vector w is approximately sufficient.

Q-learning algorithm with function approximation:

1) $\phi(s,a)$ is the feature representation for (s,a) pair. w_0 is the initial value of Q-value function parameter.

2) Repeat the following until convergence:

2.1) Initialize s_0

2.2) Repeat the following until episode termination or maximum episode step limit is reached

2.2.1 $a_t = \varepsilon\text{-greedy}(s_t)$

2.2.2 observe r_t, s_{t+1}

2.2.3

$$w_{t+1} = w_t + \alpha_t \left(r_t + \gamma \max_{a'} \phi(s_{t+1}, a')^T w_t - \phi(s_t, a_t)^T w_t \right) \phi(s_t, a_t)$$

Intuition behind the update rule:

- We want to satisfy the following bellman equation:

$$Q(s,a) = E\left[r(s,a,s') + \gamma \max_{a'} Q(s',a') \,|\, s,a \right]$$

$$\eta(\pi) = \frac{1}{2} \sum_s d^\pi(s) \sum_a \pi(a|s) \left(Q(s,a) - \phi(s,a)^T \omega \right)^2$$

$$\nabla \eta(\pi) = -\sum d^\pi(s) \sum \pi(a|s) \left(Q(s,a) - \phi(s,a)^T \omega \right) \phi(s,a)$$

$$= E\Big[\Big(E\left[r(s,a,s') + \gamma \max_{a'} Q(s',a') \,|\, s,a \right] - \phi(s,a)^T \omega \Big) \phi(s,a) \Big]$$

$$\nabla \eta(\pi) = -E\Big[r(s,a,s') + \gamma \max_{a'} \phi(s',a')^T \omega - \phi(s,a)^T \omega \Big) \phi(s,a) \Big]$$

$$\omega_{t+1} = \omega_t - \alpha_t \nabla \eta(\pi) \quad \text{// gradient descent}$$

$$w_{t+1} = w_t - \alpha_t \left(r(s_t, a_t, s_{t+1}) + \gamma \max_{a'} \phi(s_t, a')^T w_t - \phi(s_t, a_t)^T w \right) \phi(s_t, a_t)$$

// stochastic gradient descent

Remarks about Q-learning

1) It is used for infinite horizon task but can be used for finite horizon also with approximation

2) It is model free algorithm.

3) It is an off-policy algorithm

4) Tabular Q-learning is for finite space but with function approximation it can be used with infinite state space.

5) Policy is stationary

SARSA algorithm:

1) Initialise value of $Q(s,a)$ for all states and all actions.

2) Initialize S, $A = \varepsilon\text{-greedy}(S)$

- Till the episode terminates or max timestep limit is reached perform following

- Observe reward $R(S,A,S')$ and next state S'

- obtain $A' = \varepsilon\text{-greedy}(S')$

- Update Q-value:

$$Q(S,A) = Q(S,A) + \alpha\left(R(S,A,S') + \gamma\, Q(S',A') - Q(S,A)\right)$$

- $S \leftarrow S'$ $A \leftarrow A'$

3) Repeat step 2 untill convergence

SARSA algorithm with function approximation:

1) $\phi(s,a)$ is the feature representation for (s,a) pair. w_0 is the initial value of Q-value function parameter.

) Repeat the following until convergence:

2.1) Initialize s_0

2.2) Repeat the following until episode termination or maximum episode step limit is reached

2.2.1 $a_t = \varepsilon\text{-greedy}(s_t)$

2.2.2 observe r_t, s_{t+1}

2.2.3 $a_{t+1} = \varepsilon\text{-greedy}(s_{t+1})$

2.2.4

$$w_{t+1} = w_t + \alpha_t\left(r_t + \gamma\,\phi(s_{t+1}, a_{t+1})^T w_t - \phi(s_t, a_t)^T w_t\right)\phi(s_t, a_t)$$

Remarks about SARSA:

1) SARSA stands for State Action Reward State Action

2) SARSA is model free

3) Policy is stationary

4) SARSA is on-policy algorithm

Here data is generated by ϵ-greedy policy and the Q-value is also being obtained for ϵ-greedy policy.

Policy Evaluation Scheme

- Policy evaluation algorithms to be discussed in this section obtains the value / Q-value function corresponding to a policy.
- Once the value/Q-value function is obtained policy improvement step can be taken.
- Policy evaluation along with policy improvement step is used to implement policy iteration scheme.
- Policy evaluation algorithms discussed here will all be model free algorithms and online in nature.

Monte Carlo Policy Evaluation

1. We have a policy π
2. Until convergence perform the below:
   - 2.1 using π collect transitions for an episode: $S_0, A_0, R_0, S_1, A_1, R_1, \ldots S_{T-1}, A_{T-1}, R_{T-1}$
   - 2.2 t in $\{T-1, T-2, \ldots 0\}$:

$$G_t = \gamma G_t + R_t$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\left(G_t - Q(S_t, A_t)\right)$$

   - 2.3 $G_t = 0$

Intuition

- Let us consider a particular (S, a) pair

Episode 1: ---- (S, a) ---------- Discounted summation of Reward G_1

Episode 2: ---- (S, a) ---------- Discounted summation of Reward G_2

- The idea is to estimate the quantity $Q(S, a)$ as follows:

$$\hat{Q}_n(S, a) = \frac{G_1 + G_2 + \cdots G_n}{n}$$

$$\hat{Q}_{n+1}(S, a) = \frac{G_1 + G_2 + \cdots G_n + G_{n+1}}{n+1}$$

$$\hat{Q}_{n+1}(s,a) = \frac{n\hat{Q}_n(s,a) + G_{n+1}}{n+1}$$

$$\Rightarrow \hat{Q}_{n+1}(s,a) = \hat{Q}_n(s,a)\left(1 - \frac{1}{n+1}\right) + \frac{G_{n+1}}{n+1}$$

$$= \hat{Q}_n(s,a)(1-\alpha) + G_{n+1}\alpha$$

$$\boxed{\hat{Q}_{n+1}(s,a) = \hat{Q}_n(s,a) + \alpha\left(G_{n+1} - \hat{Q}_n(s,a)\right)}$$

This is finally used as update rule.

Monte Carlo Policy Evaluation with function Approximation

1. $\phi(s,a)$ is the feature vector for (s,a) pair and ω_0 is the initial parameter for q-value function

2. Repeat until convergence step below:

   2.1 Collect transitions for an episode using policy π : $S_0 A_0 R_0, S_1, A_1, R_1 - \cdots S_{T-1} A_{T-1} R_{T-1}$

   2.2 For t in $\{T-1, T-2, \cdots 0\}$

      2.2.1 $G_t = \gamma G_t + R_t$

      2.2.2

$$\omega_{K+1} = \omega_K + \alpha\left(G_t - \phi(S_t, A_t)^T \omega_K\right) \phi(S_t, A_t)$$

   2.3 $G_t = 0$

Intuition

- We want to minimise the below given objective function

$$\eta(\pi) = \frac{1}{2} E\left[\left(Q(s,a) - \phi(s,a)^T w \right)^2 \right]$$

$$\nabla_w \eta(\pi) = -E\left[\left(Q(s,a) - \phi(s,a)^T w \right) \phi(s,a) \right]$$

Gradient Descent:

$$w \leftarrow w - \alpha \nabla_w \eta(\pi)$$

Stochastic Gradient Descent:

$$w \leftarrow w + \alpha \left(Q(s,a) - \phi(s,a)^T w \right) \phi(s,a)$$

$$w \leftarrow w + \alpha \left(G_t - \phi(s,a)^T w \right) \phi(s,a)$$

Here G_t is the estimate of $Q(s,a)$.

Remarks:

1. Monte Carlo policy evaluation is for infinite horizon task.

2. It is model free method

3. We are finding Q-value for the policy used for collecting data. Hence it is on-policy method.

4. Function approximation version can be used for infinite state and action space.

5. Discounted reward criterion is considered.

Temporal Difference Learning

- In monte carlo method we observe all the rewards until the end of episode and then we are able to find estimate of $Q(s,a)$.
- We can observe a single reward and use the bellman equation to find an estimate of $Q(s,a)$. This approach of finding value/Q-value function is called TD(0).
- Bellman equation for Q-value for policy π:

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a\right]$$

$$Q^{\pi}(s,a) = E\left[r(s,a) + \gamma Q^{\pi}(s',a') \mid s,a\right]$$

TD(0) Algorithm

1. Initialise $Q(s,a)$ for all (s,a) pair

2. Repeat steps below until convergence:

   2.1 Initialize s_0, $a_0 \sim \pi(\cdot|s_0)$

   2.2 Repeat until episode termination or maximum episode step limit:

      2.2.1 Observe reward r_t and next state s_{t+1}

      2.2.2 $a_{t+1} \sim \pi(\cdot|s_{t+1})$

      2.2.3 update $Q(s_t, a_t)$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t \left(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t a_t) \right)$$

Remarks

1. The main motivation behind TD(0) algorithm is that we want to satisfy Bellman equation for Q-value.

2. The update rule for Q-value follow the same intuition as used for the update rule in tabular Q-learning.

3. TD(0) is a model free algorithm because access to reward function and transition probability is not assumed.

4. Q-value is estimated for the same policy which is used to generate data. Hence TD(0) is on-policy method.

Temporal Difference Learning with function approximation

Intuition

- $\phi(s,a)$ is the feature vector for (s,a) pair. $Q(s,a)$ is approximated by $\phi(s,a)^T w$. Here w is called the Q-value function parameter.
- We want to find a value of w parameter such that $\phi(s,a)^T w$ is the best possible approximation of the true Q-value function of policy π.
- We minimize the following objective function to find the best approximation:

$$\eta(w) = \frac{1}{2} E\left[\left(\vec{Q}(s,a) - \phi(s,a)^T w\right)^2\right]$$

Here $\vec{Q}(s,a)$ is the true Q-value.

$$\nabla \eta(\omega) = -E\left[\left(\hat{Q}^{\pi}(s,a) - \phi(s,a)^T \omega\right)\phi(s,a)\right]$$

We know that

$$Q^{\pi}(s,a) = E\left[\mathcal{R}(s,a) + \gamma\, Q^{\pi}(s',a') \,\middle|\, s,a\right]$$

$$\nabla \eta(\omega) = -E\Big[\Big(E\left[\mathcal{R}(s,a) + \gamma\, Q^{\pi}(s',a') \,\middle|\, s,a\right] - \phi(s,a)^T\omega\Big)\,\phi(s,a)\Big]$$

$$= -E\Big[\Big(\mathcal{R}(s,a) + \gamma\, Q^{\pi}(s',a') - \phi(s,a)^T\omega\Big)\phi(s,a)\Big]$$

$$\approx -E\Big[\Big(\mathcal{R}(s,a) + \gamma\, \phi(s',a')^T\omega - \phi(s,a)^T\omega\Big)\phi(s,a)\Big]$$

Because of this particular approximation step the gradient used for TD(0) is not a gradient direction.

- The stochastic gradient descent update rule for Q-value parameter w is as follows:

$$w_{t+1} = w_t + \alpha_t \Big(\big(r(s_t, a_t) + \phi(s_{t+1}, a_{t+1})^T w_t - \phi(s_t, a_t)^T w_t \big) \phi(s_t, a_t) \Big)$$

TD(0) with Function Approximation

1. Initialise Q-value parameter w_0.

2. Repeat steps below until convergence:

   2.1 Initialize s_0, $a_0 \sim \pi(\cdot|s_0)$

   2.2 Repeat until episode termination or maximum episode step limit:

      2.2.1 Observe reward r_t and next state s_{t+1}

      2.2.2 $a_{t+1} \sim \pi(\cdot|s_{t+1})$

      2.2.3 update w_t as:

$$w_{t+1} = w_t + \alpha_t \Big(\big(r(s_t, a_t) + \gamma \phi(s_{t+1}, a_{t+1})^T w_t - \phi(s_t, a_t)^T w \big) \phi(s_t, a_t) \Big)$$

Remarks:

1) Q-value is estimated for the same policy that is used for data collection. Hence it is on-policy algorithm

2) Because of function approximation very large state space can be handled.

3) On-policy TD(0) is convergent but off-policy TD(0) diverges for certain tasks. Ex: Baird's test.

4) In place of off-policy TD(0), Gradient Temporal Difference Learning algorithm can be used for off-policy case.

Connection between Monte Carlo and TD(0) algorithm

1 step TD

$$Q(s_0, a_0) = E\left[r(s_0, a_0) + \gamma\, Q(s_1, a_1) \,\middle|\, s_0, a_0 \right]$$

n step TD

$$Q(s_0, a_0) = E\left[\sum_{i=0}^{n-1} \gamma^i r(s_i, a_i) + \gamma^n Q(s_n, a_n) \,\middle|\, s_0, a_0 \right]$$

∞-step TD

$$Q(s_0, a_0) = E\left[\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \,\middle|\, s_0, a_0 \right]$$

Note:

The bellman equation in 1 step TD is used for TD(0) algorithm and the equation in ∞-step TD is used for Monte Carlo algorithm.

Conclusion

1) Covered value iteration based model free algorithms: Q-learning, SARSA

2) Difference between on-policy and off-policy algorithm by comparing Q-learning and SARSA.

3) Importance of function approximation

4) Discussion of model free policy evaluation algorithm (Monte Carlo and TD(0).

To be Covered next:

1) Actor Critic Algorithm

2) Deep Q-Network