Reinforcement Learning: Model Free Control

Earl Wong

So Far

Previously, we computed state value functions for two cases:

- Case1 = Model Based MDP: Apply Value Iteration => state value function V

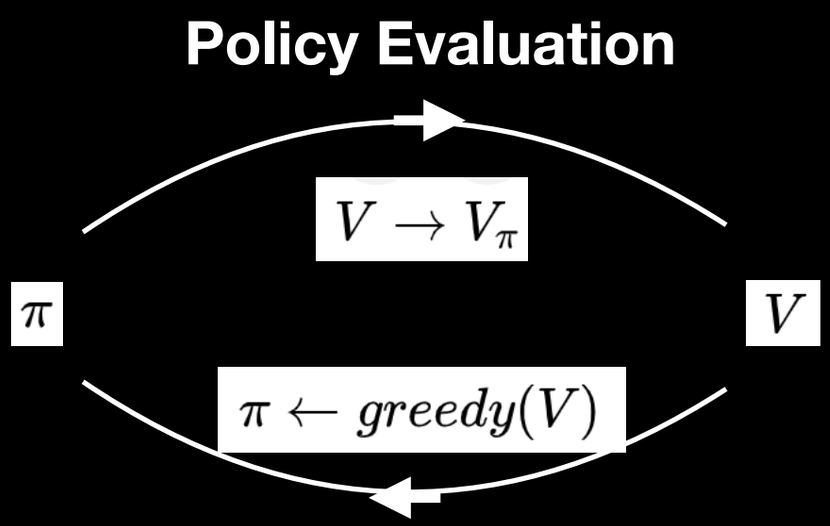
- Case2 = Model Free: Apply MC or TD Learning => state value function V

So Far

- In the model based case, we also determined an optimal policy, using policy iteration.
- Under policy iteration, we alternated between policy evaluation and policy improvement.
- In the policy evaluation step, we evaluated a known policy until some stopping condition was met.
- In the policy improvement step, we selected a new policy, where $\pi' = greedy(\pi)$

Recall: Policy Iteration

The entire algorithm was summarized as:



$$\pi_* \rightarrow V_*$$

$$\pi_* \leftarrow V_*$$

1) Initialize a value function V(s) = 0 for all states.

2) Apply a random policy.

3) Continue to apply the policy for all states, until a stopping criteria is satisfied.

4) Update the current policy, by applying a greedy policy improvement, using the current value function.

5) Repeat steps 3 and 4 until convergence / improvement stops.

This produces the optimal policy and the optimal value function.

Note: Convergence is guaranteed by the contraction mapping theorem.

Model Free Control

- Now, we would like to determine an optimal policy WITHOUT using a model.
- i.e. Without knowledge of P(s’ | s, a) and R(s, a, s’) in an MDP.
- This requires learning a policy from sample rollouts.
- i.e. Learning by trial and error.

New Concepts

To facilitate this, we introduce two new ideas:

- To compliment the state value function V(s), we introduce the action value function Q(s,a).

- To facilitate learning from sampled rollouts, we introduce “stochasticity” into the applied policy (used to create the rollout).

Previous & Now

- Previous Policy Evaluation: $V_\pi(s)$
- New Policy Evaluation: $Q_\pi(s, a)$

- Previous Policy Improvement: $\pi' = greedy(\pi)$
- New Policy Improvement: Explore the state space

New Policy Improvement

The closest policy to the greedy policy that enables exploration of the state space is:

$$\epsilon\ -\ greedy\ \ policy$$

$$A \leftarrow \begin{cases} argmax_a Q(s,a), & with\ probability\ 1-\epsilon \\ random\ \ a, & with\ probability\ \epsilon \end{cases}$$

MC Control

For every episode: $\{S_1, A_1, R_1, S_2, A_2, R_2, \ldots, S_T\} \sim \pi$

For every count: $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$

For every Q(s,a): $Q(S_t, A_t) \leftarrow \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$

For the policy improvement: $\pi \leftarrow \epsilon - greedy(Q)$

However, MC Control requires complete episodes.

Previously, temporal differencing had several advantages:
1) Allowed incomplete sequences, 2) Allowed online updates and 3) Had lower variance (but higher bias)

TD Control

For every "SARSA" timestep:

$$\{S, A, R, S', A'\} \sim \pi$$

Update Q(s,a):

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

For the policy improvement:

$$\pi \leftarrow \epsilon - greedy(Q)$$

Online SARSA Algorithm

Initialize Q(S,A) randomly, Q(terminal,) = 0

repeat (for each episode)
Initialize S
Choose A from S, using the policy derived from Q (epsilon greedy)

repeat (for each step of the episode)
Take action A
Observe R and S'
Choose A' from S' using the policy derived from Q (epsilon greedy)

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

$$S \leftarrow S', \;\; A \leftarrow A'$$

until S is terminal

Recall: Multistep TD and $TD(\lambda)$

- In general, for various n, the TD target is:
- N = 0 $G_t^0 = R_t + \gamma V(S_{t+1} = s')$
- N = 1 $G_t^1 = R_t + \gamma R_{t+1} + \gamma^2 V(S_{t+2} = s')$
- N = n $G_t^n = R_t + \gamma R_{t+1} + \ldots + \gamma^n R_{t+n} + \gamma^{n+1} V(S_{t+n+1} = s')$
- If N = n =T corresponds to the monte carlo rollout / episode, we can then combine the multistep TD results, producing $TD(\lambda)$

$$G_t^\lambda = (1-\lambda) \sum_{i=0}^{n} \lambda^n G_t^i$$

$$V(S_t = s) \leftarrow V(S_t = s) + \alpha(G_t^\lambda - V(S_t = s))$$

We Can Compute a Similar $TD(\lambda)$ Quantity for Q

- In general, for various n, the TD target is:
- N = 0 $G_t^0 = R_t + \gamma Q(S_{t+1} = s', A_{t+1} = a')$
- N = 1 $G_t^1 = R_t + \gamma R_{t+1}^2 + \gamma^2 Q(S_{t+2} = s', A_{t+2} = a')$
- N = n $G_t^n = R_t + \gamma R_{t+1} + \ldots + \gamma^{n+1} Q(S_{t+n+1} = s',\ A_{t+n+1} = a')$
- If N = n =T corresponds to the monte carlo rollout / episode, we can then combine the multistep TD results, producing $TD(\lambda)$

$$G_t^\lambda = (1-\lambda)\sum_{i=0}^{n} \lambda^n G_t^i$$

$$Q(S_t = s, A_t = a) \leftarrow Q(S_t = s, A_t = a) + \alpha(G_t^\lambda - Q(S_t = s,\ A_t = a))$$

Recall

- MC and $TD(\lambda)$ required the entire episode to be observed, before an update could occur.
- In practice, a backward view of $TD(\lambda)$ also existed, where online updates can occur at every time step, using incomplete sequences.
- This backward view involved computing eligibility traces.
- Qualitatively, eligibility traces track the frequency and the recency of a state visit.
- This information is then incorporated into the update.

Eligibility Traces

Using eligibility traces E, let:

$$E_0(s, a) = 0$$

$$E_t(s, a) = \lambda E_{t-1}(s, a) + Indicator(S_t = s, A_t = a)$$

Where λ tracks the recency of a {state, action} visit.

The TD error can then be written as:

$$\delta_t = R_t + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

Incorporating the eligibility trace into the update equation, we obtain:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t E_t(s, a)$$

Sarsa λ Algorithm

Initialize Q(S,A) randomly

repeat (for each episode)
Initialize S, A

repeat (for each step of the episode)
Take action A
Observe R and S'
Choose A' from S' using the policy derived from Q (epsilon greedy)

$$\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$$

$$E(S, A) \leftarrow E(S, A) + 1$$

$$for\ all\ \ s \in S, a \in A(s):$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t E_t(s, a)$$

$$E(s, a) \leftarrow \lambda E(s, a)$$

$$S \leftarrow S',\ \ A \leftarrow A'$$

until S is terminal

Online vs Offline and On Policy vs Off Policy

- Online means that you are learning by interacting with the environment, while offline means you are learning from a fixed dataset.

- By construction, exploration is not possible in the offline case. (However, the fixed dataset may have resulted from a highly exploratory behavior policy.)

- On policy means that you are learning about the same policy that you are using to act with in the environment.

- Off policy means that you are learning a different policy (target policy) than the one used to generate the fixed dataset (behavior policy).

Online vs Offline and On Policy vs Off Policy

- By construction, the slide titled TD Control and Online SARSA algorithm described an online, on policy approach.
- i.e. The policy interacts with the environment.

- However, you can also have an offline, off policy algorithm such as offline Q-learning.

- i.e. The policy does not interact with the environment. You are learning a policy from data generated by a different policy.

Offline, Off Policy Q Learning

A behavior policy μ is used to select specific actions.

i.e. μ is used to choose the action A in the {S, A, R, S', A'} sequence.

The fixed dataset returns a reward R and a new state S'.

The subsequent action A' is then drawn from the target policy π .

The Q function is then updated in the usual temporal difference fashion.

$$A_t \sim u(\cdot|S_t)$$

$$A' \sim \tau(\cdot|S_t)$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Offline, Off Policy Q Learning

In practice, the actions chosen from the target policy π are the actions that produces the maximum Q value for the given state S'.

The TD update equation then becomes:

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

This results in the offline, off policy Q Learning algorithm.

Offline, Off Policy Q Learning

Initialize Q(S,A) randomly, Q(terminal,) = 0

repeat (for each episode)
Initialize S

repeat (for each step of the episode)
Choose A from S using the behavior policy (epsilon greedy)
Take action A (based on the dataset result)
Observe R and S' (based on the dataset result)

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

$$S \leftarrow S'$$

until S is terminal

Importance Sampling

- Although we have two different policies - a behavior policy and a target policy, we are currently using them in a disjoint fashion.

- In future algorithms, the information from both policies can be used to “track” each other.

- Hence, we now “tease” the concept of importance sampling.

Importance Sampling

Importance sampling allow the calculation of the expectation of a function f(x) using a given distribution p(x), with a new distribution q(x).

Mathematically, this is what happens:

$$E_{X \sim p}[f(x)] = \sum_x p(x) f(x)$$

$$= \sum_x q(x) \frac{p(x)}{q(x)} f(x)$$

$$= E_{X \sim q}\left[\frac{p(x)}{q(x)} f(x)\right]$$

In off policy RL, we typically have samples collected from a behavior policy u (denoted as q above), but would like to estimate expectations under a target policy π denoted as p above.

Importance Sampling and Off Policy MC

This concept can be used to update the value function:

$$G_t^{\frac{\pi}{\mu}} = \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \frac{\pi(A_{t+2}|S_{t+2})}{\mu(A_{t+2}|S_{t+2})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\frac{\pi}{\mu}} - V(S_t))$$

Unfortunately, good updates will not result. Why?

The product of the target policies is high variance.

The product of the behavior policies is high variance.

The ratio of two high variance quantities is even higher variance.

Hence, the updates for the value function are very high variance, and are useless.

Importance Sampling and Off Policy TD

By using (only) the first term in the ratio, a significantly lower variance update results.

Ideally, we would like the two policies to be fairly similar over a single step, resulting in numerical updates that do not differ by many orders of magnitude.

$$V(S_t) \leftarrow V(S_t) + \alpha \ \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_t + \gamma V(S_{t+1}) \ - V(S_t))$$