Model Free Control

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

September 09, 2023

Overview

Towards Model Free Control

Problem and Motivation

- **Goal** : How can we learn a good policy?
- **Motivation** : Many real world applications can be modelled as MDP
  - ★ Games like Backgammon and Go
  - ★ Robot Locomotion
  - ★ Inventory or supply chain management
- For almost all these problems, model is unknown or computationally infeasible; but sampling experiences is possible
- Learning better policies through experiences is model free control

Towards Model Free Control

DP algorithms for control

- Value Iteration
- Policy Iteration

Question : Can we do model free control with value iteration ?

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

- Value iteration may not come in handy because it requires knowledge of model; so not suitable

Question : How about policy iteration (PI) ? ?

- PI is a two step provess
  - ★ Policy evaluation
  - ★ Policy improvement

Policy Iteration : Recap

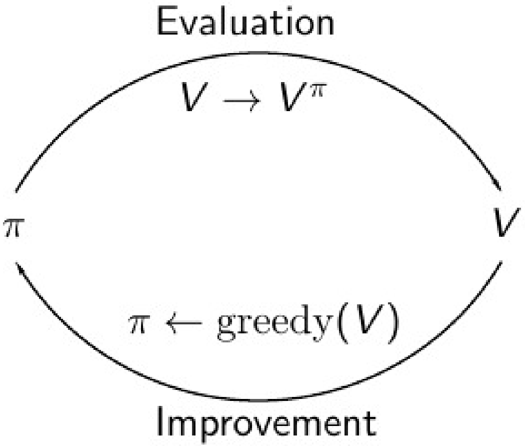


Figure Source: Sutton and Barto

On Policy Improvement From Samples

- (Greedy) Policy improvement

$$\pi(s) \quad = \quad \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]$$

- Generally, model free control is not done with V as greedy policy improvement over V requires the knowledge of the model
- (Greedy) policy improvement over Q is model free

$$\pi(s) = \arg\max_a Q^\pi(s, a)$$

- For model-free policy improvement, we use Q^π, not V^π

Core Idea behind Model Free Control

- Initialize a policy π
- Repeat
  - ★ Policy Evaluation : Find Q^{π}
  - ★ Policy Improvement : Get an improved policy from evaluation of Q^{π}

Monte Carlo Control

Policy Evaluation : Action Value Function

- We now need to evaluate Q^π instead of V^π
- Recall that the state-action value function of a policy π is given by,

$$
\begin{aligned}
Q^\pi(s,a) &\stackrel{\text{def}}{=} \mathbb{E}_\pi(G_t|s_t = s, a_t = a) \\
&= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right) \\
&= \mathbb{E}_\pi(r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a)
\end{aligned}
$$

- We can use MC or TD methods to evaluate Q^π using samples

First Visit Monte Carlo : Action Value Function

- To evaluate $Q^\pi(s,a)$ for some given state s and action a, repeat over several episodes
  - ★ The first time t that $s_t = s$ and $\pi(s) = a$ in the episode
    1. Increment counter for number of visits to s: $N(s,a) \leftarrow N(s,a) + 1$
    2. Increment running sum of total returns with return from current episode: $S(s,a) \leftarrow S(s,a) + G_t$
- Monte Carlo estimate of value function $Q(s,a) \leftarrow S(s,a)/N(s,a)$

The main drawback of this algorithm is

- Many state action pairs may never be visited
- If policy π is deterministic, things get even worse

Exploring Starts Assumption

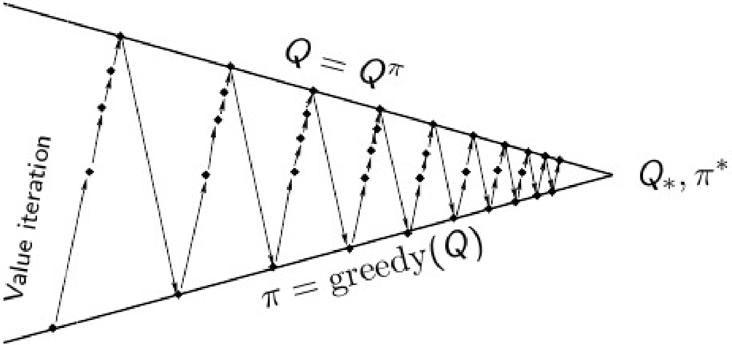
Exploring Starts (ES) Assumption

- First step of each episode start at a state-action pair, and that every such pair has non-zero probability of being selected at start
- Guarantees that all state-action pairs will be visited an infinite number of times in the limit of an infinite number of episodes

Not a realistic assumption at all !! But let's assume it for a while

- With ES assumption, first or every visit MC algorithm will evaluate Q^{π}

Policy Iteration with Action Value Function



- Monte Carlo Policy Evaluation, $Q = Q^{\pi}$
- Greedy policy improvement , $\pi' = \arg\max_a Q^{\pi}(s, a)$

Figure Source: Sutton and Barto

Monte Carlo Control with ES

Algorithm Monte Carlo Control with ES

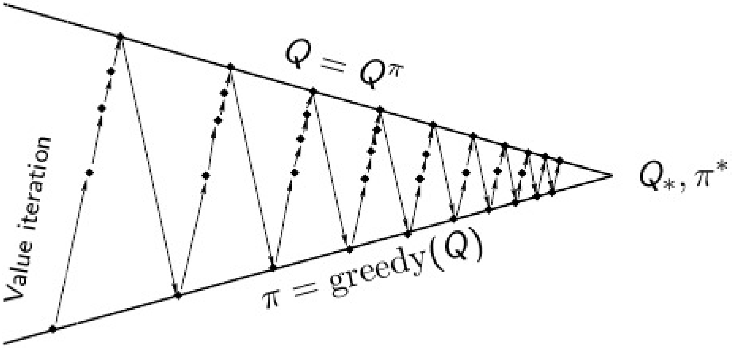
1: Start with an initial policy π_1;
2: **for** $k = 1, 2, \cdots, K$ **do**
3: Policy Evaluation Step : Evaluate Q^{π_k} using first or every visit MC
4: Policy Improvement Step :

$$\pi_{k+1} = \arg\max_a Q^{\pi_k}(s, a)$$

5: **end for**

- Convergence of policy evaluation to Q^{π} is assured only under the ES assumption
- Once ES assumption is made, to understand convergence to Q_* and π_* one can use the same kind of arguments as we had in the policy iteration algorithm in the DP setting

Policy Iteration with Action Value Function



- Is it good to be always greedy ?
- Should we patiently wait until policy evaluation step converges ?

 Figure Source: Sutton and Barto

On Greedy Action Selection

- There are two doors in front of you
- You open the left door and get reward 0 i.e. $V(\text{left}) = 0$
- You open the right door and get reward 1 $V(\text{right}) = 1$
- You open the right door and get reward 3 $V(\text{right}) = 2$
- You open the right door and get reward 2 $V(\text{right}) = 3$
- Are we sure that right door is the best door ?

ε-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability every time
  - ★ With probability $1-\varepsilon$, choose the greedy action
  - ★ With probability ε, choose an action uniformly at random

$$
\begin{aligned}
\pi(a|s) &= \frac{\varepsilon}{m} + 1 - \varepsilon, \text{ if } \ a = \arg\max_{a'} Q(s,a'), \\
&= \frac{\varepsilon}{m}, \text{ otherwise}
\end{aligned}
$$

ε-Greedy Policy Improvement

For any policy ε-greedy policy π, the ε-greedy policy π' w.r.t. Q^π is an improvement over π, that is, $V^{\pi'}(s) \geq V^\pi(s)$

$\varepsilon-$ Greedy Policy Improvement

$$
\begin{aligned}
Q^{\pi}(s,\pi'(s)) &= \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m}\sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon)\max_a Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m}\sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon)\frac{1-\varepsilon}{1-\varepsilon}\max_a Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m}\sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon)\sum_a \frac{\pi(a|s)-\frac{\varepsilon}{m}}{1-\varepsilon}\max_a Q^{\pi}(s,a) \\
&\geq \frac{\varepsilon}{m}\sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon)\sum_a \frac{\pi(a|s)-\frac{\varepsilon}{m}}{1-\varepsilon} Q^{\pi}(s,a) \\
&= \sum_{a\in\mathcal{A}} \pi(a|s) Q^{\pi}(s,a) = V^{\pi}(s) \qquad (1)
\end{aligned}
$$

Therefore, $V^{\pi'}(s) \geq V^{\pi}(s)$ from the policy improvement theorem

GLIE

Definition

G*reedy in the* **L***imit with* **I***nfinite* **E***xploration*

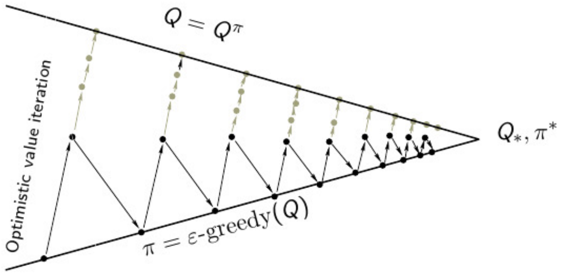
- All state-action pairs are visited infinitely often
- The policy converges to a purely greedy policy

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}_{a = \arg\max_{a'} Q_k(s,a)}$$

- ε-greedy is GLIE if ε decays to 0 asymptotically, for example,

$$\varepsilon_k = \frac{1}{k}$$

Optimistic GLIE Policy Iteration



Every episode

- Monte Carlo Policy Evaluation $Q \approx Q^\pi$
- Policy improvement using ϵ- greedy with ε decay

Figure Source: Sutton and Barto

GLIE Monte Carlo Control

Algorithm Monte Carlo Control : GLIE

Initalize Q(s,a) = 0, set $\varepsilon = 1$;
Create an ε-greedy initial policy π_1;
for $k = 1, 2, \cdots, K$ **do**
  Sample a trajectory from policy π_k
  for For each state action (s_t, a_t) pair in the trajectory **do**
    Compute the total discounted return G_t starting from (s_t, a_t)

$$N(s_t, a_t) = N(s_t, a_t) + 1$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G_t - Q(s_t, a_t))$$

  end for
  Set $\epsilon \leftarrow \frac{1}{k}$ and perform the policy improvement step as

$$\pi_{k+1} = \epsilon\text{-greedy}(\pi_k)$$

end for

TD Control

TD Control

- Natural idea : Use TD instead of MC in policy iteration framework
- Apply TD to evaluate $Q(s, a)$ in the evaluation step
- Use ε-greedy policy improvement in the update step

TD Evaluation of Q Function

- State-action value function of a policy π:

$$\begin{aligned} Q^\pi(s,a) &\overset{\text{def}}{=} \mathbb{E}_\pi(G_t | s_t = s, a_t = a) \\ &= \mathbb{E}_\pi(r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a) \end{aligned}$$

- Iterative DP policy evaluation:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \sum_{a'} (\pi(s',a') Q_k(s',a')) \right]$$

$$Q_k \to Q^\pi$$

- TD approximation: Given the transition $(s_t, a_t, r_{t+1}, s_{t+1})$, sample $a' \sim \pi(s_{t+1}, \cdot)$, and update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t [r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t)]$$

TD Evaluation of Q Function : SARSA

- TD approximation: Given the transition $(s_t, a_t, r_{t+1}, s_{t+1})$, sample $a' \sim \pi(s_{t+1}, \cdot)$, and perform the following update

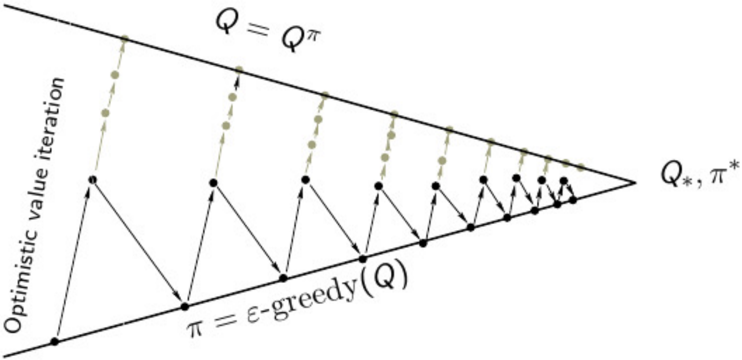
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t)]$$

- On-policy version (SARSA): $a_t \sim \pi(s_t, \cdot)$
- Off-policy version: $a_t \sim \mu(s_t, \cdot)$;
  - ★ Need to multiply the term inside square brackets with suitable importance sampling factor

TD Evaluation : Convergence

- On Policy and off-policy version covnerges to Q^π
  - ★ Convergence takes place under similar conditions as TD methods for V^π
    - State and action spaces are finite
    - All state-action pairs are visited infinitely often
    - Robbins-Monroe condition: $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$

Optimistic Policy Iteration



Along every episode, we interleave one step of policy evaluation followed ϵ-greedy policy improvement

Figure Source: Sutton and Barto

SARSA: On-Policy Control

- Policy is always ε-greedy with ε decay
- Given a trajectory segment $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ generated by the ε-greedy policy, update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Algorithm SARSA

Initialize $Q(s, a)$ arbitrarily, with Q at terminal states set to zero
for Repeat for each episode **do**
    Initialize s, choose action a at s using ϵ-greedy over Q
    for Repeat for each step in the episode **do**
        Take action a, observe reward r and next state s'
        Choose action a' for state s' using ϵ-greedy over Q

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)], s \leftarrow s', a \leftarrow a'$$

    end for
end for

Learning Optimal State-Action Value Function

- Optimal Q function:

$$Q_*(s,a) \stackrel{\text{def}}{=} \max_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$$

- Bellman optimality equation:

$$Q_*(s,a) \quad = \quad \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q_*(s_{t+1},a') | s_t = s, a_t = a\right]$$

- Iterative DP approximation

$$Q_{k+1}(s,a) \quad \leftarrow \quad \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_k(s',a')\right]$$

$$Q_k \rightarrow Q_*$$

Q-Learning: Off-Policy Control

- Policy is always ε-greedy with ε decay
- Given a trajectory segment $(s_t, a_t, r_{t+1}, s_{t+1})$ generated by the ε-greedy policy, update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

Algorithm Q-Learning

Initialize $Q(s, a)$ arbitrarily, with Q at terminal states set to zero
for Repeat for each episode **do**
  Initialize s, choose action a at s using ϵ-greedy over Q
  for Repeat for each step in the episode **do**
    Take action a, observe reward r and next state s'
    Choose target to update $Q(s, a)$ by being greedy at s' as shown below

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], s \leftarrow s'$$

  end for
end for

SARSA and Q-Learning : Backup diagram

- Q-learning is an off-policy algorithm
  - Target policy is greedy w.r.t to $Q(s,a)$,
  - Behaviour policy is ε-greedy w.r.t to $Q(s,a)$

Backup Diagrams for SARSA and Q-Learning

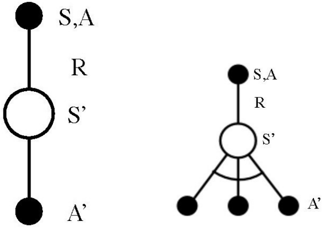


Figure Source: Sutton and Barto

Summary and Closing Remarks

Summary

- MC-based evaluation of V^π (also possible for Q^π)
- GLIE Monte-Carlo control converges to optimal action value function
- TD-based approximate evaluation of V^π, Q^π
  - 1-step TD, n-step TD, TD(λ), SARSA
  - Convergence guarantees under infinite exploration, and Robbins-Monroe condition
- TD-based control
  - On-policy control with SARSA (also possible: n-step SARSA, SARSA(λ))
  - Off-policy control with Q-learning
  - Based on optimistic policy iteration, and GLIE

MC Vs TD Control

- TD methods have several advantages over MC methods
  - Lower variance
  - Online
  - Partial sequences

Schematic View of MC and TD Algorithms

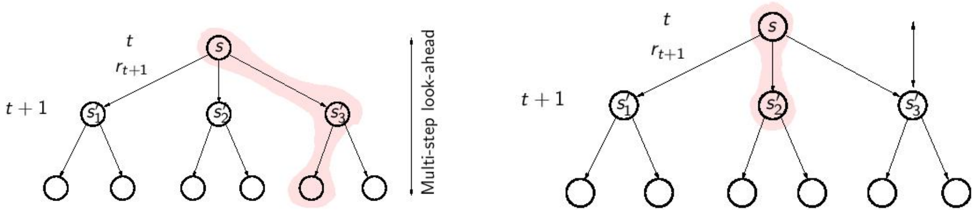
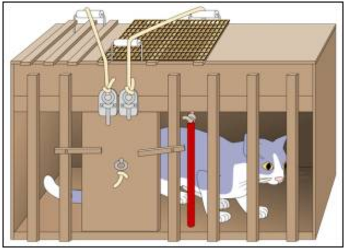
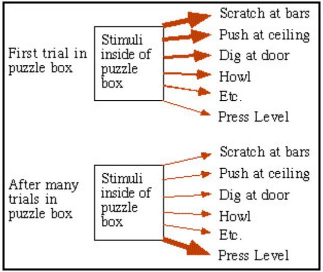


Figure: MC Algorithm and TD Algorithms

Thondrike's Cat and Exploration



Thondrike's cat

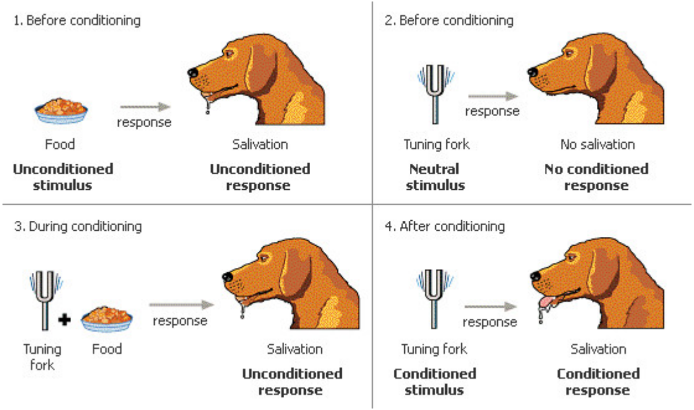


Actions by cat

ϵ-greedy strategy helps to explore !!

Figure Source: Oscar Education : Blogpost

Pavlov's Dog and Temporal Difference

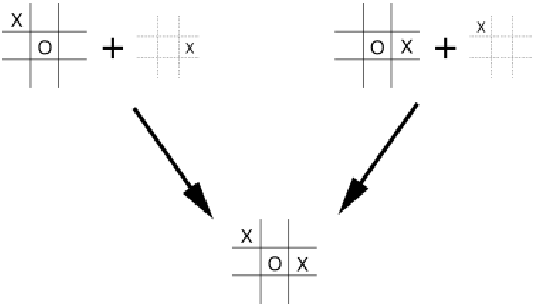


Pavlov's Dog

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$$

Figure Source: https://www.age-of-the-sage.org/psychology/pavlov.html

Afterstates



- Tic-Tac-Toe : States : Board positions and moves are actions
- A conventional action-value function ($Q(s, a)$) would map or learn about the two state action pairs on the top row separately
- An afterstate value function would immediately evaluate both equally
- Any learning about the position-move pair on the left would immediately transfer to the pair on the right

 Figure Source: Sutton and Barto

What Next ?

All methods discussed under model free methods are in the tabular setting

- Next: richer ways to represent value functions
- Needed for very large (or continuous) state spaces
- What if the action space is large (or continuous)?

Over to Deep RL !!