Model-free RL with Value Function Approximation Continued

Introduction to Reinforcement Learning

Bogdan Ivanyuk-Skulskyi, Dmytro Kuzmenko

Department of Mathematics,
National University of Kyiv-Mohyla Academy

March 13, 2023

Table of Contents

Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

- Define the mean squared error of a linear value function approximation for a particular policy relative to the true value as

$$MSVE_{\mu}(w) = \sum_{s \in S} \mu(s)(V^{\pi}(s) - \hat{V}(s; w))^2$$

- where
  - $\mu(s)$: probability of visiting state s under policy π. Note $\sum_s \mu(s) = 1$
  - $\hat{V}^{\pi}(s; w) = x(s)^T w$, a linear value function approximation
- Monte Carlo policy evaluation with VFA converges to the weights w_{MC} which has the minimum mean squared error possible with respect to the distribution μ:

$$MSVE_{\mu}(w) = min_{w} \sum_{s \in S} \mu(s)(V^{\pi}(s) - \hat{V}(s; w))^2$$

Convergence Guarantees for TD Linear VFA for Policy Evaluation: Preliminaries

- For infinite horizon, the Markov Chain defined by a MDP with a particular policy will eventually converge to a probability distribution over states $d(s)$
- $d(s)$ is called the stationary distribution over states of π
- $\sum_s d(s) = 1$
- $d(s)$ satisfies the following balance equation:

$$d(s') = \sum_s \sum_a \pi(a|s)\pi(s'|s,a)d(s)$$

Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

- Define the mean squared error of a linear value function approximation for a particular policy relative to the true value given the distribution d as

$$MSVE_d(w) = \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}(s; w))^2$$

- where
  - $d(s)$: stationary distribution of π in the true decision process.
  - $\hat{V}^\pi(s; w) = x(s)^T w$, a linear value function approximation
- TD(0) policy evaluation with VFA converges to weights w_{TD} which is within a constant factor of the min mean squared error possible given distribution d:

$$MSVE_d(w_{TD}) = \frac{1}{1-\gamma} min_w \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; w))^2$$

Table of Contents

Recall Incremental Model-Free Control Approaches

- Similar to policy evaluation, true state-action value function for a state is unknown and so substitute a target value
- In Monte Carlo methods, use a return Gt as a substitute target

$$\Delta w = \alpha(G_t - \hat{Q}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

- For SARSA instead use a TD target $r + \gamma \hat{Q}(s', a'; w)$ which leverages the current function approximation value

$$\Delta w = \alpha(r + \gamma \hat{Q}(s', a'; w) - \hat{Q}(s, a; w))\nabla_w \hat{Q}(s, a; w)$$

- For Q-learning instead use a TD target $r + \gamma max_{a'} \hat{Q}(s', a'; w)$ which leverages the max of the current function approximation value

$$\Delta w = \alpha(r + \gamma max_{a'} \hat{Q}(s', a'; w) - \hat{Q}(s, a; w))\nabla_w \hat{Q}(s, a; w)$$

Convergence of TD Methods with VFA

- Informally, updates involve doing an (approximate) Bellman backup followed by best trying to fit underlying value function to a particular feature representation
- Bellman operators are contractions, but value function approximation fitting can be an expansion

Active Area: Off Policy Learning with Function Approximation

- Extensive work in better TD-style algorithms with value function approximation, some with convergence guarantees: see Chp 11 SB

Value Function Approximation

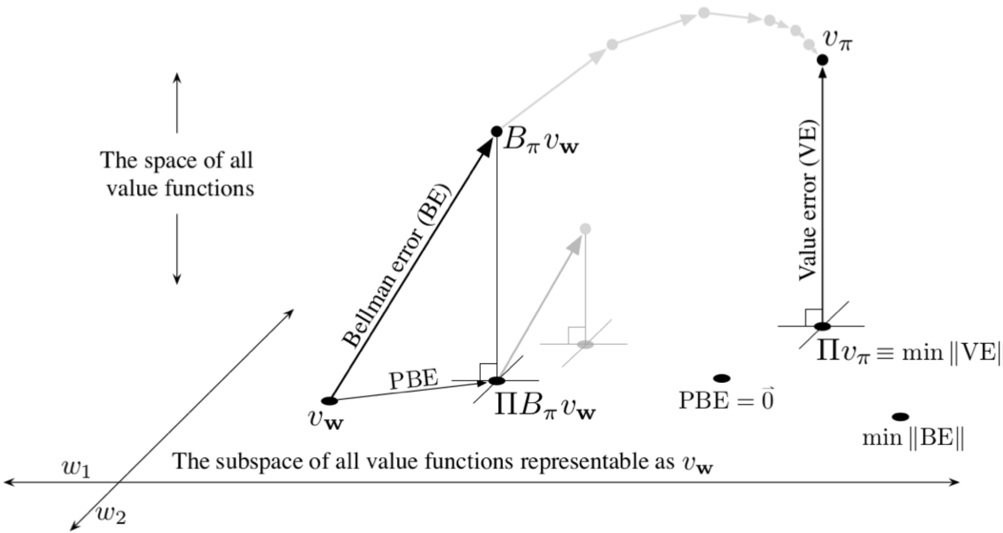


Table of Contents

Maximization Bias Proof

- Consider single-state MDP ($|S| = 1$) with 2 actions, and both actions have 0-mean random rewards, ($\mathbb{E}(r|a = a_1) = \mathbb{E}(r|a = a_2) = 0$).
- Then $Q(s, a_1) = Q(s, a_2) = 0 = V(s)$
- Assume there are prior samples of taking action a_1 and a_2
- Let $\hat{Q}(s, a_1)$, $\hat{Q}(s, a_2)$ be the finite sample estimate of Q
- Use an unbiased estimator for Q: e.g. $\hat{Q}(s, a_1) = \frac{1}{n(s,a_1)} \sum_{i=1}^{n(s,a_1)} r_i(s, a1)$
- Let $\hat{\pi} = argmax_a \hat{Q}(s, a)$ be the greedy policy w.r.t. the estimated $\hat{Q}$
- Even though each estimate of the state-action values is unbiased, the estimate of $\hat{\pi}$'s value $\hat{V}^{\hat{\pi}}$ can be biased:

$$\hat{V}^{\hat{\pi}}(s) = \mathbb{E}[max \hat{Q}(s, a_1), \hat{Q}(s, a_2)] \geq max[\mathbb{E}[\hat{Q}(s, a_1)], [\hat{Q}(s, a_2)]] = max[0, 0] = V^{\pi},$$

where the inequality comes from Jensen's inequality.

Table of Contents

Double Q-Learning

- The greedy policy w.r.t. estimated Q values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of $Q_1(s_1, a_i)$ and $Q_2(s_1, a_i) \forall a$.
  - Use one estimate to select max action: $a^* = argmaxa_{Q1}(s_1, a)$
  - Use other estimate to estimate value of $a^* : Q)_2(s, a^*)$
  - Yields unbiased estimate: $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$

Double Q-Learning

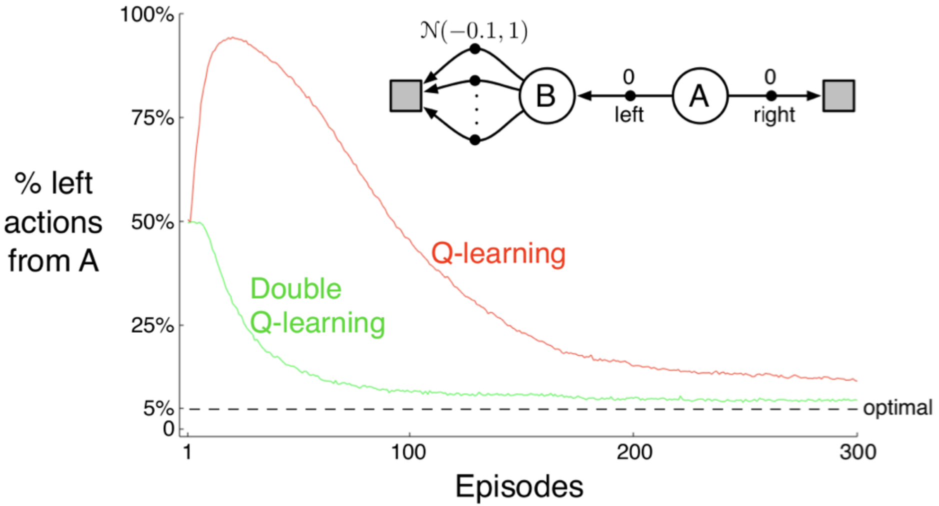
- The greedy policy w.r.t. estimated Q values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of $Q_1(s_1, a_i)$ and $Q_2(s_1, a_i) \forall a$.
  - Use one estimate to select max action: $a^* = argmaxa_{Q1}(s_1, a)$
  - Use other estimate to estimate value of $a^* : Q)_2(s, a^*)$
  - Yields unbiased estimate: $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$
- Why is this an unbiased estimate of the max state-action value?
  Using independent samples to estimate the value
- If acting online, can alternate samples used to update Q_1 and Q_2, using the other to select the action chosen

Double Q-Learning

- Initialize $Q_1(s,a)$ and $Q_2(s,a)$, $\forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
- loop
  - Select at using ϵ-greedy $\pi(s) = argmax_a Q_1(s_t, a) + Q_2(s_t, a)$
  - Observe (r_t, s_{t+1})
  - if (with prob. 0.5) then:
    - $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, argmax_a Q_1(s_{t+1}, a)) - Q_1(s_t, a_t))$
  - else
    - $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, argmax_a Q_2(s_{t+1}, a)) - Q_2(s_t, a_t))$
  - end if
  - $t = t + 1$

Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?
Doubles the memory, same computation requirements, data requirements are subtle– might reduce amount of exploration needed due to lower bias

DOUBLE Q-LEARNING (FIGURE 6.7 IN SUTTON AND BARTO 2018)



Due to the maximization bias, Q-learning spends much more time selecting suboptimal actions than double Q-learning.

Table of Contents

Recall DQN

- Deep Q-learning (DQN): Q-learning with deep neural networks and
  - Experience replay
  - Fixed Q-targets

$$\Delta w = \alpha(r + \gamma max_{a'} \hat{Q}(s', a'; w) - \hat{Q}(s, a; w)) \nabla_w \hat{Q}(s, a; w)$$

Double DQN

- Double DQN (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
- Extend double Q learning to DQN
- Current Q-network w is used to select actions
- Older Q-network w is used to evaluate actions

$$\Delta w = \alpha(r + \gamma \hat{Q}(argmax_{a'} \hat{Q}(s', a'; w); w^{-}) - \hat{Q}(s, a; w))$$

Double DQN

- Double DQN (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
- Extend double Q learning to DQN
- Current Q-network w is used to select actions
- Older Q-network w is used to evaluate actions

$$\Delta w = \alpha(r + \gamma \hat{Q}(argmax_{a'} \hat{Q}(s', a'; w); w^{-}) - \hat{Q}(s, a; w))$$

- In DQN the same weights w were used to choose the best action at s' and evaluate its value $\hat{Q}(s', a'; w^{-})$

Model-free value function approximation RL: What You Should Know

- Be able to derive weight update for generic function approximation for Q/V^{π}
- Understand various (MC/SARSA/Q-learning) targets used when updating Q function
- Know what TD vs MC converge to for policy evaluation with a linear function approximator
- Be able to implement DQN
- Define the maximization bias and give one tool for alleviating it