Lecture 4: Model Free Control and Function Approximation

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2023

- Structure and content drawn in part from David Silver's Lecture 5 and Lecture 6. For additional reading please see SB Sections 5.2-5.4, 6.4, 6.5, 6.7

Check Your Understanding L4N1: Model-free Generalized Policy Improvement

- Consider policy iteration
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement $\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s,a)$
- Question: is this π_{i+1} deterministic or stochastic?
- Answer: Deterministic, Stochastic, Not Sure
- Now consider evaluating the policy of this new π_{i+1}. Recall in model-free policy evaluation, we estimated V^π, using π to generate new trajectories
- Question: Can we compute $Q^{\pi_{i+1}}(s,a)\ \forall s,a$ by using this π_{i+1} to generate new trajectories?
- Answer: True, False, Not Sure

what if the s here can always choose det policy

$\pi_{i+1}(s)$

Check Your Understanding L4N1: Model-free Generalized Policy Improvement

- Consider policy iteration
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement $\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a)$
- Question: is this π_{i+1} deterministic or stochastic?
  Answer: Deterministic

- Now consider evaluating the policy of this new π_{i+1}. Recall in model-free policy evaluation, we estimated V^π, using π to generate new trajectories
- Question: Can we compute $Q^{\pi_{i+1}}(s, a)\ \forall s, a$ by using this π_{i+1} to generate new trajectories?
  Answer: No.

Class Structure

- Last time: Policy evaluation with no knowledge of how the world works (MDP model not given)
- Control (making decisions) without a model of how the world works
- Generalization – Value function approximation

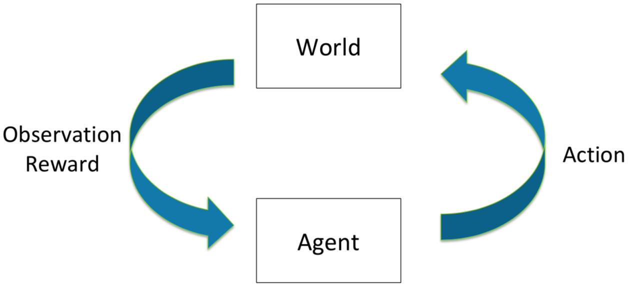
Todays Lecture

1. Model-Free Control with a Tabular Representation
   - Generalized Policy Improvement
   - Monte-Carlo Control with Tabular Representations
   - Temporal Difference Methods for Control
2. Value Function Approximation
   - Model Free Value Function Approximation Policy Evaluation
   - Monte Carlo Value Function Approximation Policy Evaluation
   - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
   - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation
3. Control using Value Function Approximation

Model-free Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^{π}
  - Policy improvement: update π given Q^{π}
- May need to modify policy evaluation:
  - If π is deterministic, can't compute $Q(s, a)$ for any $a \neq \pi(s)$
- How to interleave policy evaluation and improvement?
  - Policy improvement is now using an estimated Q

The Problem of Exploration



- Goal: Learn to select actions to maximize total expected future reward
- Problem: Can't learn about actions without trying them (need to *explore*
- Problem: But if we try new actions, spending less time taking actions that our past experience suggests will yield high reward (need to *exploit* knowledge of domain to achieve high rewards)

ϵ-greedy Policies

- Simple idea to balance exploration and achieving rewards
- Let $|A|$ be the number of actions
- Then an ϵ-greedy policy w.r.t. a state-action value $Q(s, a)$ is
  $\pi(a|s) =$ $1-\epsilon$ $\arg\max_a Q(s,a)$
  ϵ randomly select action $a \in A$

ϵ-greedy Policies

- Simple idea to balance exploration and achieving rewards
- Let $|A|$ be the number of actions
- Then an ϵ-greedy policy w.r.t. a state-action value $Q(s,a)$ is $\pi(a|s) =$
    - $\arg\max_a Q(s,a)$, w. prob $1 - \epsilon + \frac{\epsilon}{|A|}$
    - $a' \neq \arg\max Q(s,a)$ w. prob $\frac{\epsilon}{|A|}$

Policy Improvement with ϵ-greedy policies

- Recall we proved that policy iteration using given dynamics and reward models, was guaranteed to monotonically improve
- That proof assumed policy improvement output a deterministic policy
- Same property holds for ϵ-greedy policies

Monotonic ϵ-greedy Policy Improvement

Theorem

For any ϵ-greedy policy π_i, the ϵ-greedy policy w.r.t. Q^{π_i}, π_{i+1} is a monotonic improvement $V^{\pi_{i+1}} \geq V^{\pi_i}$ ← compute exactly

$$
\begin{aligned}
Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s,a) \\
&= (\epsilon/|A|) \left[\sum_{a \in A} Q^{\pi_i}(s,a) \right] + (1-\epsilon) \max_a Q^{\pi_i}(s,a)
\end{aligned}
$$

In class I left this as an optional exercise. These post lecture slides include a proof on next slide

Monotonic ϵ-greedy Policy Improvement

Theorem

For any ϵ-greedy policy π_i, the ϵ-greedy policy w.r.t. Q^{π_i}, π_{i+1} is a monotonic improvement $V^{\pi_{i+1}} \geq V^{\pi_i}$

$$
\begin{aligned}
Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s,a) \\
&= (\epsilon/|A|) \left[\sum_{a \in A} Q^{\pi_i}(s,a) \right] + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \\
&= (\epsilon/|A|) \left[\sum_{a \in A} Q^{\pi_i}(s,a) \right] + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \frac{1-\epsilon}{1-\epsilon} \\
&= (\epsilon/|A|) \left[\sum_{a \in A} Q^{\pi_i}(s,a) \right] + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1-\epsilon} \\
&\geq \frac{\epsilon}{|A|} \left[\sum_{a \in A} Q^{\pi_i}(s,a) \right] + (1-\epsilon) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1-\epsilon} Q^{\pi_i}(s,a) \\
&= \sum_{a \in A} \pi_i(a|s) Q^{\pi_i}(s,a) = V^{\pi_i}(s)
\end{aligned}
$$

Table of Contents

1. Model-Free Control with a Tabular Representation
   - Generalized Policy Improvement
   - Monte-Carlo Control with Tabular Representations
   - Temporal Difference Methods for Control
   - Model Free Value Function Approximation Policy Evaluation
   - Monte Carlo Value Function Approximation Policy Evaluation
   - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
   - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

Recall Monte Carlo Policy Evaluation, **Now for Q**

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s')$$

Initialize $Q(s,a) = 0$, $N(s,a) = 0$ $\forall (s,a)$, $k = 1$, Input $\epsilon = 1$, π ε-greedy
loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π
  Compute $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$ $\forall t$
  for $t = 1, \ldots, T$ **do**
    if First visit to **(s,a)** in episode k **then**
      $N(s,a) = N(s,a) + 1$
      $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
    end if
  end for
  $k = k + 1$
end loop

Monte Carlo Online Control / On Policy Improvement

Initialize $Q(s,a) = 0, N(s,a) = 0 \ \forall(s,a)$, Set $\epsilon = 1$, $k = 1$
$\pi_k = \epsilon\text{-greedy}(Q)$ // Create initial ϵ-greedy policy

$1-\epsilon$ prob argmax $Q(s,a)$
else random

loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
  $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$
  for $t = 1, \ldots, T$ **do**
    if First visit to (s,a) in episode k **then**
      $N(s,a) = N(s,a) + 1$
      $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
    end if
  end for
  $k = k + 1$, $\epsilon = 1/k$
  $\pi_k = \epsilon\text{-greedy}(Q)$ // Policy improvement
end loop

Properties of MC control with ϵ-greedy policies

- Computational complexity?
- Converge to optimal Q^* function?
- Empirical performance?

L4N2 Check Your Understanding: Monte Carlo Online Control / On Policy Improvement

Initialize $Q(s,a) = 0, N(s,a) = 0 \ \forall(s,a)$, Set $\epsilon = 1,\ k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
  $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$
  for $t = 1, \ldots, T$ **do**
    if First visit to (s,a) in episode k **then**
      $N(s,a) = N(s,a) + 1$
      $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
    end if
  end for
  $k = k + 1,\ \epsilon = 1/k$
  $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

behavior policy

- Is Q and estimate of Q^{π_k}? When might this procedure fail to compute the optimal Q^*?

Greedy in the Limit of Infinite Exploration (GLIE)

Definition of GLIE

- All state-action pairs are visited an infinite number of times

$$\lim_{i \to \infty} N_i(s, a) \to \infty$$

- Behavior policy (policy used to act in the world) converges to greedy policy
  $\lim_{i \to \infty} \pi(a|s) \to \underbrace{\arg\max_a Q(s, a)}$ with probability 1

$\epsilon \to 0$

Greedy in the Limit of Infinite Exploration (GLIE)

Definition of GLIE

- All state-action pairs are visited an infinite number of times

$$\lim_{i \to \infty} N_i(s, a) \to \infty$$

- Behavior policy (policy used to act in the world) converges to greedy policy
  $\lim_{i \to \infty} \pi(a|s) \to \arg\max_a Q(s, a)$ with probability 1

- A simple GLIE strategy is ϵ-greedy where ϵ is reduced to 0 with the following rate: $\epsilon_i = 1/i$

GLIE Monte-Carlo Control

Theorem

GLIE Monte-Carlo control converges to the optimal state-action value function $Q(s, a) \rightarrow Q^*(s, a)$

Table of Contents

1. Model-Free Control with a Tabular Representation
   - Generalized Policy Improvement
   - Monte-Carlo Control with Tabular Representations
   - Temporal Difference Methods for Control
   - Model Free Value Function Approximation Policy Evaluation
   - Monte Carlo Value Function Approximation Policy Evaluation
   - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
   - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

Model-free Policy Iteration with TD Methods

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^{π} using temporal difference updating with ϵ-greedy policy
  - Policy improvement: Same as Monte carlo policy improvement, set π to ϵ-greedy (Q^{π})
- First consider SARSA, which is an on-policy algorithm.
- On policy: SARSA is trying to compute an estimate Q of the policy being followed.

state action reward (next) state (next) action

General Form of SARSA Algorithm

(TD(0))

Set initial ϵ-greedy policy π randomly, $t = 0$, initial state $s_t = s_0$

Markov ✓

Take $a_t \sim \pi(s_t)$

Observe (r_t, s_{t+1})

loop

  Take action $a_{t+1} \sim \pi(s_{t+1})$ // Sample action from policy

  Observe (r_{t+1}, s_{t+2})

  Update Q given $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \big(\underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1})}_{\text{target}} - Q(s_t, a_t)\big)$$

  Perform policy improvement:

$\forall s \quad \pi(s) = \arg\max Q(s,a)$ with prob $1-\epsilon$

else random

  $t = t + 1$, $\epsilon = 1/t$

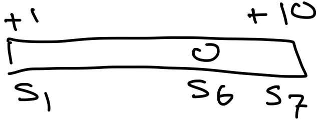
end loop

General Form of SARSA Algorithm

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_{t+1}, s_{t+2})
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$, $\epsilon = 1/t$ for ex.
end loop

Worked Example: SARSA for Mars Rover

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
  Take action $a_{t+1} \sim \pi(s_{t+1})$
  Observe (r_{t+1}, s_{t+2})
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
  $t = t + 1$
end loop



for all states

- Initialize $\epsilon = 1/k$, $k = 1$, and $\alpha = 0.5$, $Q(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $Q(-, a_2) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
  s1 s7
- Assume starting state is s_6 and sample a_1

$(s_6, a_1, 0, \underbrace{s_7, a_2})$

$Q(s_6, a_1) = (1-\alpha) \cdot 0 + \alpha\,(0 + \gamma\, Q(s_7, a_2))$

$= .5 \cdot (0 + \gamma \cdot 5) = 2.5$

$r + \gamma r' + \gamma^2 V(s'')$

Worked Example: SARSA for Mars Rover

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_{t+1}, s_{t+2})
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$
end loop

- Initialize $\epsilon = 1/k$, $k = 1$, and $\alpha = 0.5$, $Q(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $Q(-, a_2) =[\ 1\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- Tuple: $(s_6, a_1, 0, s_7, a_2, 5, s_7)$.
- $Q(s_6, a_1) = .5 * 0 + .5 * (0 + \gamma Q(s_7, a_2)) = 2.5$

Properties of SARSA with ϵ-greedy policies

- Computational complexity?
- Converge to optimal Q^* function? Recall:
  - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  - $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
  - *Q is an estimate of the performance of a policy that may be changing at each time step*
- Empirical performance?

Convergence Properties of SARSA

Theorem

SARSA for finite-state and finite-action MDPs converges to the optimal action-value, $Q(s,a) \to Q^*(s,a)$, under the following conditions:

1. The policy sequence $\pi_t(a|s)$ satisfies the condition of GLIE
2. The step-sizes α_t satisfy the Robbins-Munro sequence such that

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

- For ex. $\alpha_t = \frac{1}{T}$ satisfies the above condition.

Properties of SARSA with ϵ-greedy policies

$$\Delta_{t+1}(x) = (1-\alpha_t(x))\Delta_t(x) + \alpha_t(x) F_t(x)$$

1992, 1994

- Result builds on stochastic approximation
- Relies on step sizes decreasing at the right rate] Robbins Munro
- Relies on Bellman backup contraction property
- Relies on bounded rewards and value function

if $r \in [0, R_{max}]$ $\quad V \in \left[0, \frac{R_{max}}{1-\gamma}\right]$

In class I discussed how these sort of results rely on stochastic approximation. In particular there is a proof that Q-learning (see slides later in this lecture) will converge under a set of assumptions that leverages a

On and Off-Policy Learning

- On-policy learning
  - Direct experience
  - Learn to estimate and evaluate a policy from experience obtained from following that policy
- Off-policy learning
  - Learn to estimate and evaluate a policy using experience gathered from following a different policy

Q-Learning: Learning the Optimal State-Action Value

- SARSA is an **on-policy** learning algorithm
- SARSA estimates the value of the current **behavior** policy (policy using to take actions in the world)
- And then updates that (behavior) policy
- Alternatively, can we directly estimate the value of π^* while acting with another behavior policy π_b?
- Yes! Q-learning, an **off-policy** RL algorithm

Q-Learning: Learning the Optimal State-Action Value

- SARSA is an **on-policy** learning algorithm
- SARSA estimates the value of the current **behavior** policy (policy using to take actions in the world)
- And then updates the policy trying to estimate
- Alternatively, can we directly estimate the value of π^* while acting with another behavior policy π_b?
- Yes! Q-learning, an **off-policy** RL algorithm
- Key idea: Maintain state-action Q estimates and use to bootstrap– use the value of the best future action
- Recall SARSA

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

$$V^* = \max_a Q^*(s, a)$$

- Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

Q-Learning with ϵ-greedy Exploration

Initialize $Q(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$

Set π_b to be ϵ-greedy w.r.t. Q

(s, a, r, s')

loop

  Take $a_t \sim \pi_b(s_t)$ // Sample action from policy

  Observe (r_t, s_{t+1})

  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\overbrace{r_t + \gamma \max_a Q(s_{t+1}, a)}^{\text{style TD(0) update}} - Q(s_t, a_t))$

  $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random

  $t = t + 1$

end loop

Worked Example: ϵ-greedy Q-Learning Mars

Initialize $Q(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
Set π_b to be ϵ-greedy w.r.t. Q
loop
  Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
  Observe (r_t, s_{t+1})
  $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \overbrace{\max_a Q(s_{t+1},a)} - Q(s_t,a_t))$
  $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
  $t = t+1$
end loop

- Initialize $\epsilon = 1/k$, $k=1$, and $\alpha = 0.5$, $Q(-,a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $Q(-,a_2) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- Like in SARSA example, start in s_6 and take a_1. $s_6, a_1, 0, s_7$

$\max_a Q(s_7, a) = 10$

Worked Example: ϵ-greedy Q-Learning Mars

1: Initialize $Q(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
2: Set π_b to be ϵ-greedy w.r.t. Q
3: **loop**
4: Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
5: Observe (r_t, s_{t+1})
6: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t))$
7: $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
8: $t = t+1$
9: **end loop**

- Initialize $\epsilon = 1/k$, $k = 1$, and $\alpha = 0.5$, $Q(-, a_1) =$ [1 0 0 0 0 0 +10], $Q(-, a_2) =$[1 0 0 0 0 0 +5], $\gamma = 1$
- Tuple: $(s_6, a_1, 0, s_7)$.
- $Q(s_6, a_1) = 0 + .5 * (0 + \gamma \max_{a'} Q(s_7, a') - 0) = .5*10 = 5$
- Recall that in the SARSA update we saw $Q(s_6, a_1) = 2.5$ because we used the actual action taken at s_7 instead of the max
- Does how Q is initialized matter (initially? asymptotically?)? Asymptotically no, under mild condiditions, but at the beginning, yes

Q-Learning with ϵ-greedy Exploration

- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal Q^*?
  Visit all (s, a) pairs infinitely often, and the step-sizes α_t satisfy the Robbins-Munro sequence. Note: the algorithm does not have to be greedy in the limit of infinite exploration (GLIE) to satisfy this (could keep ϵ large).
- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal π^*?
  The algorithm is GLIE, along with the above requirement to ensure the Q value estimates converge to the optimal Q.

Table of Contents

1 Model-Free Control with a Tabular Representation
- Generalized Policy Improvement
- Monte-Carlo Control with Tabular Representations
- Temporal Difference Methods for Control

2 Value Function Approximation
- Model Free Value Function Approximation Policy Evaluation
- Monte Carlo Value Function Approximation Policy Evaluation
- Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
- Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

3 Control using Value Function Approximation

Motivation for Function Approximation

- Don't want to have to explicitly store or learn for every single state a
  - Dynamics or reward model
  - Value
  - State-action value
  - Policy
- Want more compact representation that generalizes across state or states and actions

Benefits of Function Approximation

- Reduce memory needed to store $(P, R)/V/Q/\pi$
- Reduce computation needed to compute $(P, R)/V/Q/\pi$
- Reduce experience needed to find a good $P, R/V/Q/\pi$

Function Approximators

- Many possible function approximators including
  - Linear combinations of features
  - Neural networks
  - Decision trees
  - Nearest neighbors
  - Fourier/ wavelet bases
- In this class we will focus on function approximators that are differentiable (Why?)
- Two very popular classes of differentiable function approximators
  - Linear feature representations (Today)
  - Neural networks (Next lecture)

Review: Gradient Descent

- Consider a function $J(\boldsymbol{w})$ that is a differentiable function of a parameter vector $\boldsymbol{w}$
- Goal is to find parameter $\boldsymbol{w}$ that minimizes J
- The gradient of $J(\boldsymbol{w})$ is

$$\nabla J(w) = \left[\frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial w_2} \quad \cdots \quad \frac{\partial J}{\partial w_n} \right]$$

$$w \leftarrow w - \Delta w$$

Value Function Approximation for Policy Evaluation with an Oracle

- First assume we could query any state s and an oracle would return the true value for $V^\pi(s)$
- Similar to supervised learning: assume given $(s, V^\pi(s))$ pairs
- The objective is to find the best approximate representation of V^π given a particular parameterized function $\hat{V}(s; w)$

Stochastic Gradient Descent

- Goal: Find the parameter vector $\boldsymbol{w}$ that minimizes the loss between a true value function $V^\pi(s)$ and its approximation $\hat{V}(s; \boldsymbol{w})$ as represented with a particular function class parameterized by $\boldsymbol{w}$.
- Generally use mean squared error and define the loss as

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; \boldsymbol{w}))^2]$$

- Can use gradient descent to find a local minimum

$$\Delta \boldsymbol{w} = -\frac{1}{2}\alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Stochastic gradient descent (SGD) uses a finite number of (often one) samples to compute an approximate gradient:

$$\nabla_w J(w) = \nabla_w E_\pi [V^\pi(s) - \hat{V}(s,w)]^2$$
$$= E_\pi\, 2(V^\pi(s) - \hat{V}(s,w)) \nabla_w V(s,w)$$

- Expected SGD is the same as the full gradient update

Stochastic Gradient Descent

- Goal: Find the parameter vector $\boldsymbol{w}$ that minimizes the loss between a true value function $V^\pi(s)$ and its approximation $\hat{V}(s;\boldsymbol{w})$ as represented with a particular function class parameterized by $\boldsymbol{w}$.
- Generally use mean squared error and define the loss as

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s;\boldsymbol{w}))^2]$$

- Can use gradient descent to find a local minimum

$$\Delta \boldsymbol{w} \;=\; -\frac{1}{2}\alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Stochastic gradient descent (SGD) uses a finite number of (often one) samples to compute an approximate gradient:

$$\begin{aligned}\Delta_w J(w) &= \overset{\nabla w}{\cancel{\Delta_w}} E_\pi[V^\pi(s) - \hat{V}(s;w)]^2 \\ &= - E_\pi[2(\underline{V^\pi(s)} - \hat{V}(s;w)] \overset{\nabla}{\cancel{\Delta}}_w \hat{V}(s,w)\end{aligned}$$

- Expected SGD is the same as the full gradient update

Table of Contents

- Generalized Policy Improvement
- Monte-Carlo Control with Tabular Representations
- Temporal Difference Methods for Control

2. Value Function Approximation
   - Model Free Value Function Approximation Policy Evaluation
   - Monte Carlo Value Function Approximation Policy Evaluation
   - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
   - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

Model Free VFA Policy Evaluation

- No oracle to tell true $V^\pi(s)$ for any state s
- Use model-free value function approximation

Model Free VFA Prediction / Policy Evaluation

- Recall model-free policy evaluation (Lecture 3)
  - Following a fixed policy π (or had access to prior data)
  - Goal is to estimate V^π and/or Q^π
- Maintained a lookup table to store estimates V^π and/or Q^π
- Updated these estimates after each episode (Monte Carlo methods) or after each step (TD methods)
- **Now: in value function approximation, change the estimate update step to include fitting the function approximator**

Feature Vectors

- Use a feature vector to represent a state s

$$\mathbf{x}(s) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \dots \\ x_n(s) \end{pmatrix}$$

features of state

ex fn on weight

Linear Value Function Approximation for Prediction With An Oracle

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features

$$\hat{V}(s;\boldsymbol{w}) = \sum_{j=1}^{n} x_j(s) w_j = \boldsymbol{x}(s)^T \boldsymbol{w}$$

- Objective function is

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \underbrace{\hat{V}(s;\boldsymbol{w})}_{x(s)^T w})^2]$$

- Recall weight update is

$$\Delta \boldsymbol{w} = -\frac{1}{2}\alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Update is:

$$\Delta w = (V^\pi(s) - x(s)^T w)\, x(s)$$

Linear Value Function Approximation for Prediction With An Oracle

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features

$$\hat{V}(s; \boldsymbol{w}) = \sum_{j=1}^{n} x_j(s) w_j = \boldsymbol{x}(s)^T \boldsymbol{w}$$

- Objective function is

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; \boldsymbol{w}))^2]$$

- Recall weight update is

$$\Delta \boldsymbol{w} = -\frac{1}{2}\alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Update is: $\Delta \boldsymbol{w} = \cancel{-\frac{1}{2}}\alpha(V^\pi(s) - \boldsymbol{x}(s)^T \boldsymbol{w})\boldsymbol{x}$
- Update = step-size × prediction error × feature value

Linear Value Function Approximation for Policy Evaluation: Plug In Estimate for $V^\pi(s)$

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features

$$\hat{V}(s; \boldsymbol{w}) = \sum_{j=1}^{n} x_j(s) w_j = \boldsymbol{x}(s)^T \boldsymbol{w}$$

- Objective function is

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; \boldsymbol{w}))^2]$$

- Recall weight update is

$$\Delta \boldsymbol{w} = -\frac{1}{2} \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Update is: $\Delta \boldsymbol{w} = \cancel{-\frac{1}{2}} \alpha (V^\pi(s) - \boldsymbol{x}(s)^T \boldsymbol{w}) \boldsymbol{x}$

Table of Contents

- Generalized Policy Improvement
- Monte-Carlo Control with Tabular Representations
- Temporal Difference Methods for Control

2. Value Function Approximation
    - Model Free Value Function Approximation Policy Evaluation
    - Monte Carlo Value Function Approximation Policy Evaluation
    - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
    - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

Monte Carlo Value Function Approximation

- Return G_t is an unbiased but noisy sample of the true expected return $V^\pi(s_t)$
- Therefore can reduce MC VFA to doing supervised learning on a set of (state,return) pairs: $\langle s_1, G_1 \rangle, \langle s_2, G_2 \rangle, \ldots, \langle s_T, G_T \rangle$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator

Monte Carlo Value Function Approximation

- Return G_t is an unbiased but noisy sample of the true expected return $V^\pi(s_t)$
- Therefore can reduce MC VFA to doing supervised learning on a set of (state,return) pairs: $\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \ldots, \langle s_T, G_T\rangle$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator
- Concretely when using linear VFA for policy evaluation

$V^\pi(s_t)$

$$
\begin{aligned}
\Delta \boldsymbol{w} &= \alpha(\underline{G_t} - \hat{V}(s_t;\boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{V}(s_t;\boldsymbol{w}) \\
&= \alpha(G_t - \hat{V}(s_t;\boldsymbol{w}))\boldsymbol{x}(s_t) \\
&= \alpha(G_t - \boldsymbol{x}(s_t)^T\boldsymbol{w})\boldsymbol{x}(s_t)
\end{aligned}
$$

- Note: G_t may be a very noisy estimate of true return

MC Linear Value Function Approximation for Policy Evaluation

Initialize w $= 0$, $k = 1$
loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,L_k})$ given π
  for $t = 1, \ldots, L_k$ **do**
    if First visit to (s) in episode k **then**
      $G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$
      Update weights:
      $w \leftarrow \alpha\left(G_T(s) - x(s)^T w\right) \underbrace{\vec{x}(s)}_{\text{vector}}$
    end if
  end for
  $k = k + 1$
end loop

Linear Value Function Approximation for Policy Evaluation: Plug In Estimate for $V^\pi(s)$

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features

$$\hat{V}(s; \boldsymbol{w}) = \sum_{j=1}^{n} x_j(s) w_j = \boldsymbol{x}(s)^T \boldsymbol{w}$$

- Objective function is

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; \boldsymbol{w}))^2]$$

- Recall weight update is

$$\Delta \boldsymbol{w} = \cancel{-\frac{1}{2}} \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- Update is: $\Delta \boldsymbol{w} = \cancel{-\frac{1}{2}} \alpha (V^\pi(s) - \boldsymbol{x}(s)^T \boldsymbol{w}) \boldsymbol{x}$

Table of Contents

- Generalized Policy Improvement
- Monte-Carlo Control with Tabular Representations
- Temporal Difference Methods for Control

2. Value Function Approximation
   - Model Free Value Function Approximation Policy Evaluation
   - Monte Carlo Value Function Approximation Policy Evaluation
   - Temporal Difference (TD(0)) Value Function Approximation Policy Evaluation
   - Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

Lecture finished here, to be continued on next lecture

Refresh Your Knowledge L4. Polleverywhere Poll

Optional (no formal poll)

- Which of the following equations express a TD update?
  1. $V(s_t) = r(s_t, a_t) + \gamma \sum_{s'} p(s'|s_t, a_t) V(s')$
  2. $V(s_t) = (1-\alpha) V(s_t) + \alpha (r(s_t, a_t) + \gamma V(s_{t+1}))$
  3. $V(s_t) = (1-\alpha) V(s_t) + \alpha \sum_{i=t}^{H} r(s_i, a_i)$
  4. $V(s_t) = (1-\alpha) V(s_t) + \alpha \max_a (r(s_t, a) + \gamma V(s_{t+1}))$
  5. Not sure
- Bootstrapping is
  1. When samples of (s,a,s') transitions are used to approximate the true expectation over next states
  2. When an estimate of the next state value is used instead of the true next state value
  3. Used in Monte-Carlo policy evaluation
  4. Not sure

Refresh Your Knowledge L4. Polleverywhere Poll

Optional (no formal poll)

- Which of the following equations express a TD update?
  True. $V(s_t) = (1-\alpha)V(s_t) + \alpha(r(s_t, a_t) + \gamma V(s_{t+1}))$

- Bootstrapping is when:
  An estimate of the next state value is used instead of the true next state value

Check Your Understanding L4N2: MC for On Policy Control

Optional (no formal poll)

- Mars rover with new actions:
  - $r(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [\ 0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- Assume current greedy $\pi(s) = a_1\ \forall s$, ϵ=.5
- Sample trajectory from ϵ-greedy policy
- Trajectory = (s_3, a_1, 0, s_2, a_2, 0, s_3, a_1, 0, s_2, a_2, 0, s_1, a_1, 1, terminal)
- First visit MC estimate of Q of each (s, a) pair?
- $Q^{\epsilon-\pi}(-, a_1) = [1\ 0\ 1\ 0\ 0\ 0\ 0]$, $Q^{\epsilon-\pi}(-, a_2) = [0\ 1\ 0\ 0\ 0\ 0\ 0]$
- What is $\pi(s) = \arg\max_a Q^{\epsilon-\pi}(s, a)\ \forall s$?
  π = [1 2 1 tie tie tie tie]

- Under the new ϵ-greedy policy, if $k = 3$, $\epsilon = 1/k$
  With probability $2/3$ choose $\pi(s)$ else choose randomly. As an example, $\pi(s_1) = a_1$ with prob $(2/3)$ else randomly choose an action. So the prob of picking a_1 will be $2/3 + (1/3) * (1/2) = 5/6$

SARSA Initialization

Optional (no formal poll)

- Mars rover with new actions:
  - $r(-, a_1) = [\ 1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [\ 0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- Initialize $\epsilon = 1/k$, $k = 1$, and $\alpha = 0.5$, $Q(-, a_1) = r(-, a_1)$, $Q(-, a_2) = r(-, a_2)$
- SARSA: $(s_6, a_1, 0, s_7, a_2, 5, s_7)$.
- Does how Q is initialized matter (initially? asymptotically?)? Asymptotically no, under mild condiditions, but at the beginning, yes