Lecture 4: Model Free Control [2]

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2018

[2] Structure closely follows much of David Silver's Lecture 5. For additional reading please see SB Sections 5.2-5.4, 6.4, 6.5, 6.7

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Class Structure

- Last time: Policy evaluation with no knowledge of how the world works (MDP model not given)
- This time: Control (making decisions) without a model of how the world works
- Next time: Value function approximation and Deep Q-learning

Evaluation to Control

- Last time: how good is a specific policy?
  - Given no access to the decision process model parameters
  - Instead have to estimate from data / experience
- Today: how can we learn a good policy?

Recall: Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

Today: Learning to Control Involves

- Optimization: Goal is to identify a policy with high expected rewards (similar to Lecture 2 on computing an optimal policy given decision process models)
- Delayed consequences: May take many time steps to evaluate whether an earlier decision was good or not
- Exploration: Necessary to try different actions to learn what actions can lead to high rewards

Today: Model-free Control

- Generalized policy improvement
- Importance of exploration
- Monte Carlo control
- Model-free control with temporal difference (SARSA, Q-learning)
- Maximization bias

Model-free Control Examples

- Many applications can be modeled as a MDP: Backgammon, Go, Robot locomation, Helicopter flight, Robocup soccer, Autonomous driving, Customer ad selection, Invasive species management, Patient treatment
- For many of these and other problems either:
  - MDP model is unknown but can be sampled
  - MDP model is known but it is computationally infeasible to use directly, except through sampling

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

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Recall Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute V^π
  - Policy improvement: update π

$$\pi'(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^\pi(s') = \arg\max_a Q^\pi(s,a) \tag{1}$$

- Now want to do the above two steps without access to the true dynamics and reward models
- Last lecture introduced methods for model-free policy evaluation

Model-free Generalized Policy Improvement

- Given an estimate $Q^{\pi_i}(s, a)\ \forall s, a$
- Update new policy

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a) \tag{2}$$

Model-free Policy Iteration

model-free
policy eval

- 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

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Policy Evaluation with Exploration

- Want to compute a model-free estimate of Q^π
- In general seems subtle
  - Need to try all (s, a) pairs but then follow π
  - Want to ensure resulting estimate Q^π is good enough so that policy improvement is a monotonic operator
- For certain classes of policies can ensure all (s,a) pairs are tried such that asymptotically Q^π converges to the true value

ϵ-greedy Policies

- Simple idea to balance exploration and exploitation
- Let $|A|$ be the number of actions
- Then an ϵ-greedy policy w.r.t. a state-action value $Q^\pi(s,a)$ is $\pi(a|s) =$

$$\begin{cases} a & \text{w. prob } \epsilon / |A| \\ \arg\max_a Q^\pi(s,a) & \text{w. prob } 1-\epsilon \end{cases}$$

Monotonic[19] 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 $V^{\pi_{i+1}} \geq V^{\pi_i}$

$$Q^{\pi}(s, \pi_{i+1}(s)) = \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s, a)$$

randomly   greedy

$$= (\epsilon/|A|) \sum_{a \in A} Q^{\pi_i}(s, a) + (1 - \epsilon) \max_a Q^{\pi_i}(s, a)$$

$= 1$ for any state

$$= \epsilon/|A| \sum_a Q^{\pi_i}(s,a) + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \frac{1-\epsilon}{1-\epsilon}$$

$$= \quad '' \quad '' \quad '' \quad \frac{1}{1-\epsilon} \sum_a \pi_i(a|s) - \epsilon/|A|$$

$$\geq \epsilon/|A| \sum_a Q^{\pi_i}(s,a) + (1-\epsilon) \frac{1}{1-\epsilon} \sum_a \left(\pi_i(a|s) - \epsilon/|A|\right) Q^{\pi_i}(s,a)$$

$$= \sum_a \pi_i(a|s) Q^{\pi_i}(s,a) = V^{\pi_i}(s)$$

- Therefore $V^{\pi_{i+1}} \geq V^{\pi_i}$ (from the policy improvement theorem)

[19] The theorem assumes that Q^{π_i} has been computed exactly.

Monotonic[21] 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 $V^{\pi_{i+1}} \geq V^{\pi}$

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

- Therefore $V^{\pi_{i+1}} \geq V^{\pi}$ (from the policy improvement theorem)

[21] The theorem assumes that Q^{π_i} has been computed exactly.

Greedy in the Limit of Infinite Exploration (GLIE)

Definition of GLIE

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

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

optimal

state-action v

- Behavior policy converges to greedy policy

$\lim_{i \to \infty} \pi(a|s) \Rightarrow \arg\max_a q(s,a)$ prob $\to$ 1

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

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Monte Carlo Online Control / On Policy Improvement

```
Initialize Q(s, a) = 0, Returns(s, a) = 0 ∀(s, a), Set ε = 1, k = 1
π_k = ε-greedy(Q) // Create initial ε-greedy policy
loop
    Sample k-th episode (s_k1, a_k1, r_k1, s_k2, ..., s_T) given π_k
    for t = 1, ..., T do
        if First visit to (s, a) in episode k then
            Append Σ_{j=t}^{T} r_kj to Returns(s_t, a_t)
            Q(s_t, a_t) = average(Returns(s_t, a_t))
        end if
    end for
    k = k + 1, ε = 1/k
    π_k = ε-greedy(Q^π) // Policy improvement
end loop
```

could use discount γ on rewards

ϵ-greedy(Q) → compute an ϵ-greedy policy

GLIE Monte-Carlo Control

Theorem

GLIE Monte-Carlo control converges to the optimal state-action value[a] function $Q(s, a) \to q(s, a)$

[a] $v(s)$ and $q(s, a)$ without any additional subscripts are used to indicate the optimal state and state-action value function, respectively.

Model-free Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement: update π given Q^π

- What about TD methods?

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Model-free Policy Iteration with TD Methods

- Use temporal difference methods for policy evaluation step
- 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)

sampling

TD learning (s, a, r, s', a')

$$V^\pi(s) = V^\pi(s) + \underset{\text{learning rate}}{\alpha} \left(\underbrace{r + \gamma V^\pi(s')}_{\text{bootstrapped}} - \underbrace{V^\pi(s)}_{\text{old estimate}} \right)$$

General Form of SARSA Algorithm

on policy

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$ $\quad Q(s,a) = 0 \;\; \forall s,a$

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

Observe (r_t, s_{t+1})

loop

  Take action $a_{t+1} \sim \pi(s_{t+1})$ $\quad$ action that would be taken

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

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

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \underbrace{\alpha}_{\text{learning rate}} \left(r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t) \right)$$

  Perform policy improvement:

$$\pi \leftarrow \epsilon\text{-greedy}(Q^\pi)$$

  $t = t + 1$

end loop

- What are the benefits to improving the policy after each step?
- What are the benefits to updating the policy less frequently?

Convergence Properties of SARSA

Theorem

Sarsa for finite-state and finite-action MDPs converges to the optimal action-value, $Q(s,a) \rightarrow q(s,a)$, 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

$$\begin{aligned}\sum_{t=1}^{\infty} \alpha_t &= \infty \\ \sum_{t=1}^{\infty} \alpha_t^2 &< \infty\end{aligned}$$

Recall: Off Policy, Policy Evaluation

- Given data from following a behavior policy π_b can we estimate the value V^{π_e} of an alternate policy π_b?
- Neat idea: can we learn about other ways to do things different than what we actually did?
- Discussed how to do this for Monte Carlo evaluation
- Used Importance Sampling
- First see how to do off policy evaluation with TD

Importance Sampling for Off Policy TD (Policy Evaluation)

- Recall the Temporal Difference (TD) algorithm which is used to incremental model-free evaluation of a policy π_b. Precisely, given a state s_t, an action a_t sampled from $\pi_b(s_t)$ and the observed reward r_t and next state s_{t+1}, TD performs the following update:

$$V^{\pi_b}(s_t) = V^{\pi_b}(s_t) + \alpha(r_t + \gamma V^{\pi_b}(s_{t+1}) - V^{\pi_b}(s_t)) \tag{3}$$

- Now want to use data generated from following π_b to estimate the value of different policy π_e, V^{π_e}
- Change TD target $r_t + \gamma V(s_{t+1})$ to weight target by single importance sample ratio
- New update:

likelihood under eval π
" " beh π

$$V^{\pi_e}(s_t) = V^{\pi_e}(s_t) + \alpha \left[\frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)} (r_t + \gamma V^{\pi_e}(s_{t+1}) - V^{\pi_e}(s_t)) \right] \tag{4}$$

Importance Sampling for Off Policy TD Cont.

$\prod_{t=1}^{T} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)}$ $\frac{1}{1/2}$

- Off Policy TD Update:

$$V^{\pi_e}(s_t) = V^{\pi_e}(s_t) + \alpha \left[\frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)} (r_t + \gamma V^{\pi_e}(s_{t+1}) - V^{\pi_e}(s_t)) \right] \quad (5)$$

- Significantly lower variance than MC IS. (Why?)

  Yes, only 1 importance weight instead of product

- Does π_b need to be the same at each time step?

  No, just need to know it

- What conditions on π_b and π_e are needed for off policy TD to converge to V^{π_e}? π_b has the same support

  $\pi_b(a|s) > 0$ whenever $\pi_e(a|s) \cdot V(s) > 0$

Q-Learning: Learning the Optimal State-Action Value

- Just saw how to use off policy TD to evaluate any particular policy π_e
- Can we estimate the value of the optimal policy π^* without knowledge of what π^* is?
- Yes! Q-learning
- Does not require importance sampling
- Key idea: Maintain state-action Q estimates and use to bootstrap– use the value of the best future action
- Recall Sarsa

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t)) \quad (6)$$

- Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t)) \quad (7)$$

Off-Policy Control Using Q-learning

- In the prior slide assumed there was some π_b used to act
- π_b determines the actual rewards received
- Now consider how to improve the behavior policy (policy improvement)
- Let behavior policy π_b be ϵ-greedy with respect to (w.r.t.) current estimate of the optimal $q(s, a)$

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
loop
    Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
    Observe (r_t, s_{t+1})
    Update Q given (s_t, a_t, r_t, s_{t+1}):

$$Q(s,a) = Q(s,a) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s,a)\right)$$

    Perform policy improvement: set π_b to be ϵ-greedy w.r.t. Q
    $t = t + 1$
end loop

- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal q? visit all s,a pairs ∞ often
- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal π^*? decay ϵ

Table of Contents

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Maximization Bias[39]

- 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
- Assume using an unbiased estimator for Q: e.g. $\hat{Q}(s, a_1) = \frac{1}{n(s,a_1)} \sum_{i=1}^{n(s,a_1)} r_i(s, a_1)$
- Let $\hat{\pi} = \underline{\arg\max_a \hat{Q}(s, a)}$ 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:

ex.
100 samples a_1
100 samples a_2

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

expe. our random → in rewards

Jensen's inequ

[39] Example from Mannor, Simester, Sun and Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. Management Science 2007

Double 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: $a^* = \arg\max_a Q_1(s_1, a)$
  - Use other estimate to estimate value of a^*: $Q_2(s, a^*)$
  - Yields unbiased estimate: $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$

a_1 a_2
Q_1 50 50
Q_2 50 50

- Why does this yield an unbiased estimate of the max state-action value?

Q_2 is an unbiased estimator of a_1 & a_2

- If acting online, can alternate samples used to update Q_1 and Q_2, using the other to select the action chosen
- Next slides extend to full MDP case (with more than 1 state)

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 a_t using ϵ-greedy $\pi(s) = \arg\max_a Q_1(s_t, a) + Q_2(s_t, a)$
    Observe (r_t, s_{t+1})
    if (with 0.5 probability) **then**
        $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha$
    else
        $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha$ $\left(r_t + \gamma Q_2\left(s_{t+1}, \max_{a'} Q_1(s_{t+1}, a')\right) - Q_2(s_t, a_t)\right)$
    end if
    $t = t + 1$
end loop

behavior policy
Q learning off policy

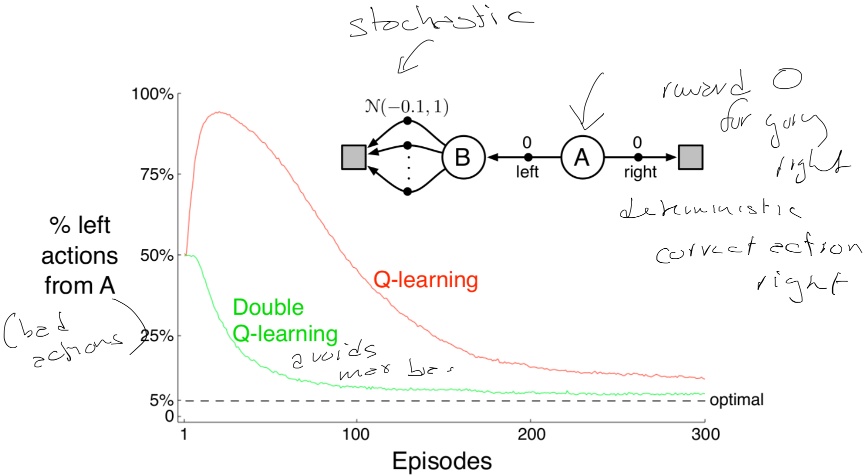
doubles

- Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?

same

unclear

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

1. Generalized Policy Iteration
2. Importance of Exploration
3. Monte Carlo Control
4. Temporal Difference Methods for Control
5. Maximization Bias

Class Structure

- Last time: Policy evaluation with no knowledge of how the world works (MDP model not given)
- This time: Control (making decisions) without a model of how the world works
- **Next time: Value function approximation and Deep Q-learning**