ECE276B: Planning & Learning in Robotics
Lecture 12: Model-free Control

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Zhichao Li: zhl355@eng.ucsd.edu

Ehsan Zobeidi: ezobeidi@eng.ucsd.edu

Ibrahim Akbar: iakbar@eng.ucsd.edu

UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Model-free Generalized Policy Iteration

- **Model-based case**: our main tool for solving a stochastic infinite-horizon problem was Generalized Policy Iteration (GPI):
  - **Policy Evaluation**: Given π, compute V^π:

$$V^\pi(x) = \ell(x, \pi(x)) + \gamma \mathbb{E}_{x' \sim p_f(\cdot|x,\pi(x))} \left[V^\pi(x') \right], \quad \forall x \in \mathcal{X}$$

  - **Policy Improvement**: Given V^π obtain a new policy π':

$$\pi'(x) = \arg\min_{u \in \mathcal{U}(x)} \underbrace{\left\{ \ell(x, u) + \gamma \mathbb{E}_{x' \sim p_f(\cdot|x,u)} \left[V^\pi(x') \right] \right\}}_{Q^\pi(x,u)}, \quad \forall x \in \mathcal{X}$$

- **Model-free case**: is it still possible to implement the GPI algorithm?
  - **Policy Evaluation**: given π, we saw in the previous lecture that MC or TD can be used to estimate V^π or Q^π
  - **Policy Improvement**: computing π' based on V^π requires access to $\ell(x, u)$ but based on Q^π can be done **without knowing** $\ell(x, u)$:

$$\pi'(x) = \arg\min_{u \in \mathcal{U}(x)} Q^\pi(x, u)$$

Policy Evaluation (Recap)

- Given π, iterate $\mathcal{T}_\pi$ to compute V^π or Q^π via Dynamic Programming (DP), Temporal Difference (TD), or Monte Carlo (MC)

- DP needs a model but TD and MC are model-free

- **Value function**:

$$DP : \mathcal{T}_\pi[V](x_t) = \ell(x_t, \pi(x_t)) + \gamma \mathbb{E}_{x_{t+1} \sim p_f(\cdot|x_t, \pi(x_t))} [V(x_{t+1})]$$

$$TD : \mathcal{T}_\pi[V](x_t) \approx V(x_t) + \alpha [\ell(x_t, u_t) + \gamma V(x_{t+1}) - V(x_t)]$$

$$MC : \mathcal{T}_\pi[V](x_t) \approx V(x_t) + \alpha \left[\sum_{k=0}^{T-t-1} \gamma^k \ell(x_{t+k}, u_{t+k}) + \gamma^{T-t} \mathfrak{q}(x_T) - V(x_t) \right]$$

- **Q function**:

$$DP : \mathcal{T}_\pi[Q](x_t, u_t) = \ell(x_t, u_t) + \gamma \mathbb{E}_{x_{t+1} \sim p_f(\cdot|x_t, u_t)} [Q(x_{t+1}, \pi(x_{t+1}))]$$

$$TD : \mathcal{T}_\pi[Q](x_t, u_t) \approx Q(x_t, u_t) + \alpha [\ell(x_t, u_t) + \gamma Q(x_{t+1}, u_{t+1}) - Q(x_t, u_t)]$$

$$MC : \mathcal{T}_\pi[Q](x_t, u_t) \approx Q(x_t, u_t) + \alpha \left[\sum_{k=0}^{T-t-1} \gamma^k \ell(x_{t+k}, u_{t+k}) + \gamma^{T-t} \mathfrak{q}(x_T) - Q(x_t, u_t) \right]$$

Model-free Policy Improvement

- If Q^π, instead of V^π, is estimated via MC or TD, the policy improvement step can be implemented model-free, i.e., can compute $\min_u Q^\pi(x, u)$ without knowing the motion model p_f or the state cost ℓ
- The fact that Q^π is an approximation to the true Q-function still causes problems:
  - Picking the "best" control according to the current estimate Q^π might not be the actual best control
  - If a deterministic policy is used for Evaluation/Improvement, one will observe returns for only one of the possible controls at each state and also might not visit many states. Hence, estimating Q^π will not be possible at those never-visited states and controls.

Example: Greedy Control Selection (David Silver)

- There are two doors in front of you
- You open the left door and get reward 0
  $\ell(left) = 0$
- You open the right door and get reward +1
  $\ell(right) = -1$
- You open the right door and get reward +3
  $\ell(right) = -3$
- You open the right door and get reward +2
  $\ell(right) = -2$
- Are you sure the right door is the best long-term choice?

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

Model-free Control

- Two ideas to ensure that you do not commit to the wrong controls too early and continue exploring the state and control spaces:

  1. **Exploring Starts**: in each episode $\rho^{(k)} \sim \pi$, choose initial state-control pairs with non-zero probability among all possible pairs $\mathcal{X} \times \mathcal{U}$

  2. ϵ-**Soft Policy**: a **stochastic policy** under which every control has a non-zero probability of being chosen and hence every reachable state will have non-zero probability of being encountered

First-visit MC Policy Iteration with Exploring Starts

Algorithm 1 MC Policy Iteration with Exploring Starts

```
Init: Q(x, u), π(x) for all x ∈ 𝒳 and u ∈ 𝒰
loop
    Choose (x_0, u_0) ∈ 𝒳 × 𝒰 randomly                                  ▷ exploring starts!
    Generate an episode ρ = x_0, u_0, x_1, u_1, ..., x_{T−1}, u_{t−1}, x_T from π
    for each x, u in ρ do
        L ← return following the first occurrence of x, u
        Q(x, u) ← Q(x, u) + α (L − Q(x, u))
    for each x in ρ do
        π(x) ← arg min_u Q(x, u)
```

ϵ-Greedy Exploration

- An alternative to exploring starts
- To ensure exploration it must be possible to encounter all $|\mathcal{U}(x)|$ controls at state x with non-zero probability
- ϵ-**Greedy Policy**: a stochastic policy that picks the best control according to $Q(x, u)$ in the policy improvement step but ensures that all other controls are selected with a small (non-zero) probability:

$$\pi(u \mid x) = \mathbb{P}(u_t = u \mid x_t = x) := \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{U}(x)|} & \text{if } u = \arg\min_{u' \in \mathcal{U}(x)} Q(x, u') \\ \frac{\epsilon}{|\mathcal{U}(x)|} & \text{otherwise} \end{cases}$$

ϵ-Greedy Policy Improvement

Theorem: ϵ-Greedy Policy Improvement

For any ϵ-soft policy π with associated Q^π, the ϵ-greedy policy π' with respect to Q^π is an improvement, i.e., $V^{\pi'}(x) \le V^\pi(x)$ for all $x \in \mathcal{X}$

- **Proof**:

$$
\begin{aligned}
\mathbb{E}_{u' \sim \pi'(\cdot|x)} \left[Q^\pi(x, u') \right] &= \sum_{u' \in \mathcal{U}(x)} \pi'(u' \mid x) Q^\pi(x, u') \\
&= \frac{\epsilon}{|\mathcal{U}(x)|} \sum_{u' \in \mathcal{U}(x)} Q^\pi(x, u') + (1 - \epsilon) \min_{u \in \mathcal{U}(x)} Q^\pi(x, u) \\
&\le \frac{\epsilon}{|\mathcal{U}(x)|} \sum_{u' \in \mathcal{U}(x)} Q^\pi(x, u') + (1 - \epsilon) \sum_{u \in \mathcal{U}(x)} \frac{\pi(u \mid x) - \frac{\epsilon}{|\mathcal{U}(x)|}}{1 - \epsilon} Q^\pi(x, u) \\
&= \sum_{u \in \mathcal{U}(x)} \pi(u \mid x) Q^\pi(x, u) = V^\pi(x)
\end{aligned}
$$

- Then, from the policy improvement theorem, $V^{\pi'}(x) \le V^\pi(x), \forall x \in \mathcal{X}$

First-visit MC Policy Iteration with ϵ-Greedy Improvement

Algorithm 2 First-visit MC Policy Iteration with ϵ-Greedy Improvement

```
Init: Q(x, u), π(u|x) (ε-soft policy) for all x ∈ 𝒳 and u ∈ 𝒰
loop
    Generate an episode ρ := x_0, u_0, x_1, u_1, ..., x_{T−1}, u_{t−1}, x_T from π
    for each x, u in ρ do
        L ← return following the first occurrence of x, u
        Q(x, u) ← Q(x, u) + α (L − Q(x, u))
    for each x in ρ do
        u* ← arg min_u Q(x, u)
        π(u|x) ← { 1 − ε + ε/|𝒰(x)|   if u = u*
                 { ε/|𝒰(x)|           if u ≠ u*
```

Init: $Q(x,u)$, $\pi(u|x)$ (ϵ-soft policy) for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$

loop

Generate an episode $\rho := x_0, u_0, x_1, u_1, \ldots, x_{T-1}, u_{t-1}, x_T$ from π

for each x, u in ρ **do**

$L \leftarrow$ return following the first occurrence of x, u

$Q(x,u) \leftarrow Q(x,u) + \alpha\,(L - Q(x,u))$

for each x in ρ **do**

$u^* \leftarrow \arg\min_u Q(x,u)$

$$\pi(u|x) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{U}(x)|} & \text{if } u = u^* \\ \frac{\epsilon}{|\mathcal{U}(x)|} & \text{if } u \neq u^* \end{cases}$$

Temporal-Difference Control

- TD prediction has several advantages over MC prediction:
  - Works with incomplete episodes
  - Can perform online updates to Q^π after every transition
  - The TD estimate of Q^π has lower variance than the MC one
- TD in the policy iteration algorithm:
  - Use TD for policy evaluation
  - Can update $Q(x, u)$ after every transition within an episode
  - Use an ϵ-greedy policy for policy improvement because we still need to trade off exploration and exploitation

TD Policy Iteration with ϵ-Greedy Improvement (SARSA)

- **SARSA**: estimates the action-value function Q^π using TD updates after every $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$ transition:

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \alpha \left[\ell(x_t, u_t) + \gamma Q(x_{t+1}, u_{t+1}) - Q(x_t, u_t)\right]$$

- Ensures exploration via an ϵ-greedy policy in the policy improvement step

Algorithm 3 SARSA

Init: $Q(x, u)$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}$
loop
    $\pi \leftarrow \epsilon$-greedy policy derived from Q
    Generate episode $\rho := (x_{0:T}, u_{0:T-1})$ from π
    for $(x, u, x', u') \in \rho$ **do**
        $Q(x, u) \leftarrow Q(x, u) + \alpha \left[\ell(x, u) + \gamma Q(x', u') - Q(x, u)\right]$

Convergence of Model-free Policy Iteration

- **Greedy in the Limit with Infinite Exploitation** (GLIE):
  - All state-control pairs are explored infinitely many times: $\lim_{k\to\infty} N(x,u) = \infty$
  - The ϵ-greedy policy converges to a greedy policy

$$\lim_{k\to\infty} \pi_k(u \mid x) = \mathbb{1}\{u = \arg\min_{u' \in \mathcal{U}(x)} Q(x,a')\}$$

- Example: If $\epsilon_k = \frac{1}{k}$, then ϵ-greedy is GLIE

Theorem: Convergence of Model-free Policy Iteration

Both MC Policy Iteration and SARSA converge to the optimal action-value function, $Q(x,u) \to Q^*(x,u)$, as the number of episodes $k \to \infty$ as long as:

- the sequence of ϵ-greedy policies $\pi_k(u \mid x)$ is GLIE,
- the sequence of step sizes α_k is Robbins-Monro.

On-Policy vs Off-Policy Learning

- **On-policy Prediction**: estimate V^π or Q^π using experience from π
- **Off-policy Prediction**: estimate V^π or Q^π using experience from μ
- On-policy methods:
  - evaluate or improve the policy π that is used to make decisions and collect experience
  - require well-designed exploration functions
  - empirically successful with function approximation
- Off-policy methods:
  - evaluate or improve a policy π that is different from the (behavior) policy μ used to generate data
  - can use an effective exploratory policy μ to generate data while learning about an optimal policy
  - can learn from observing other agents (or humans)
  - can re-use experience from old policies $\pi_1, \pi_2, \ldots, \pi_{k-1}$
  - can learn about multiple policies while following one policy
  - have problems with function approximation and eligibility traces

Importance Sampling for Off-policy Learning

- Off-policy learning: use returns generated from μ to evaluate π
- The stage costs obtained from μ, need to be re-weighted according to the similarity (i.e., likelihood) of the states encountered by π
- **Importance Sampling**: estimates the expectation of a function with respect to a different distribution:

$$\begin{aligned}\mathbb{E}_{x\sim p(\cdot)}[f(x)] &= \int p(x)f(x)dx\\ &= \int q(x)\frac{p(x)}{q(x)}f(x)dx = \mathbb{E}_{x\sim q(\cdot)}\left[\frac{p(x)}{q(x)}f(x)\right]\end{aligned}$$

Importance Sampling for Off-policy MC Learning

- To use returns generated from μ to evaluate π via MC, weight the long-term cost L_t via importance-sampling corrections along the whole episode:

$$L_t^{\pi/\mu} = \frac{\pi(u_t|x_t)}{\mu(u_t|x_t)} \frac{\pi(u_{t+1}|x_{t+1})}{\mu(u_{t+1}|x_{t+1})} \cdots \frac{\pi(u_{T-1}|x_{T-1})}{\mu(u_{T-1}|x_{T-1})} L_t$$

- Update the value estimate towards the *corrected return*:

$$V(x_t) \leftarrow V(x_t) + \alpha \left(L_t^{\pi/\mu} - V(x_t) \right)$$

- Importance sampling in MC can dramatically increase the variance and cannot be used if μ is zero when π is non-zero

Importance Sampling for Off-policy TD Learning

- To use returns generated from μ to evaluate π via TD, weight the TD target $\ell(x,u)+\gamma V(x')$ by importance sampling:

$$V(x_t) \leftarrow V(x_t) + \alpha \left(\frac{\pi(u_t \mid x_t)}{\mu(u_t \mid x_t)} \left(\ell(x_t, u_t) + \gamma V(x_{t+1})\right) - V(x_t) \right)$$

- Importance sampling in TD is much lower variance than in MC and the policies need to be similar (i.e., μ should not be zero when π is non-zero) over a single step only

Off-policy TD Control without Importance Sampling

- **Q-Learning** (Watkins, 1989): one of the early breakthroughs in reinforcement learning was the development of an off-policy TD algorithm that does not use importance sampling

- Q-Learning approximates $\mathcal{T}_*[Q](x,u)$ directly using samples:

$$Q(x_t,u_t) \leftarrow Q(x_t,u_t) + \alpha \left[\ell(x_t,u_t) + \gamma \min_{u \in \mathcal{U}(x_{t+1})} Q(x_{t+1},u) - Q(x_t,u_t) \right]$$

- The learned Q function eventually approximates Q^* **regardless of the policy being followed**!

Theorem: Convergence of Q-Learning

Q-Learning converges almost surely to Q^* assuming all state-control pairs continue to be updated and the sequence of step sizes α_k is Robbins-Monro.

- C. J. Watkins and P. Dayan. "Q-learning," Machine learning, 1992.

Q-Learning: Off-policy TD Learning

Algorithm 4 Q-Learning

Init: $Q(x, u)$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}$
loop
  $\pi \leftarrow \epsilon$-greedy policy derived from Q
  Generate episode $\rho := (x_{0:T}, u_{0:T-1})$ from π
  for $(x, u, x') \in \rho$ **do**
    $Q(x, u) \leftarrow Q(x, u) + \alpha \left[\ell(x, u) + \gamma \min_{u'} Q(x', u') - Q(x, u)\right]$

Relationship Between Full and Sample Backups

Full Backups (DP)	**Sample Backups** (TD)
Policy Evaluation $V(x) \leftarrow \mathcal{T}_\pi[V](x) = \ell(x, \pi(x)) + \gamma \mathbb{E}_{x'}\left[V(x')\right]$	**TD Prediction** $V(x) \leftarrow V(x) + \alpha(\ell(x,u) + \gamma V(x') - V(x))$
Policy Q-Evaluation $Q(x,u) \leftarrow \mathcal{T}_\pi[Q](x,u) = \ell(x,u) + \gamma \mathbb{E}_{x'}\left[Q(x', \pi(x'))\right]$	**TD Prediction Step in SARSA** $Q(x,u) \leftarrow Q(x,u) + \alpha(\ell(x,u) + \gamma Q(x',u') - Q(x,u))$
Value Iteration $V(x) \leftarrow \mathcal{T}_*[V](x) = \min_u \left\{\ell(x,u) + \gamma \mathbb{E}_{x'}\left[V(x')\right]\right\}$	**N/A**
Q-Value Iteration $Q(x,u) \leftarrow \mathcal{T}_*[Q](x,u) = \ell(x,u) + \gamma \mathbb{E}_{x'}\left[\min_{u'} Q(x',u')\right]$	**Q-Learning** $Q(x,u) \leftarrow Q(x,u) + \alpha\left(\ell(x,u) + \gamma \min_{u'} Q(x',u') - Q(x,u)\right)$

Batch Sampling-based Q-Value Iteration

Algorithm 5 Batch Sampling-based Q-Value Iteration

Init: $Q_0(x, u)$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}$
loop
  $\pi \leftarrow \epsilon$-greedy policy derived from Q
  Generate episodes $\{\rho^{(k)}\}_{k=1}^{K}$ from π
  for $(x, u) \in \mathcal{X} \times \mathcal{U}$ **do**

$$Q_{i+1}(x, u) = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{t=0}^{T} \mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) \mathbb{1}\{(x_t^{(k)}, u_t^{(k)}) = (x, u)\}}{\sum_{t=0}^{T} \mathbb{1}\{(x_t^{(k)}, u_t^{(k)}) = (x, u)\}}$$

- Batch Sampling-based Q-Value Iteration behaves like $Q_{i+1} = \mathcal{T}_*[Q_i] + \text{noise}$. Does it actually converge?

Least-squares Backup Version

- $Q_{i+1}(x,u) = \textbf{mean}\left\{\mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}),\ \forall k, t \text{ such that } (x_t^{(k)}, u_t^{(k)}) = (x,u)\right\}$
- Note that: $\textbf{mean}\left\{x^{(k)}\right\} = \arg\min_x \sum_{k=1}^K \|x^{(k)} - x\|^2$
- $$Q_{i+1}(x,u) = \arg\min_q \sum_{k=1}^K \sum_{(x_t^{(k)}, u_t^{(k)})=(x,u)} \left\|\mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) - q\right\|^2$$
- $$Q_{i+1} = \arg\min_Q \sum_{k=1}^K \sum_{t=0}^T \left\|\mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) - Q(x_t^{(k)}, u_t^{(k)})\right\|^2$$

Algorithm 6 Batch Least-squares Q-Value Iteration

1: **Init**: $Q_0(x,u)$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}$
2: **loop**
3: $\pi \leftarrow \epsilon$-greedy policy derived from Q
4: Generate episodes $\{\rho^{(k)}\}_{k=1}^K$ from π
5: $$Q_{i+1} = \arg\min_Q \sum_{k=1}^K \sum_{t=0}^T \left\|\mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) - Q(x_t^{(k)}, u_t^{(k)})\right\|^2$$

Small Steps in the Backup Direction

- Full backup: $Q_{i+1} \leftarrow \mathcal{T}_*[Q_i] + \text{noise}$
- Partial backup: $Q_{i+1} \leftarrow \alpha\mathcal{T}_*[Q_i] + (1-\alpha)Q_i + \text{noise}$
- Equivalent to a gradient step on squared error objective function:

$$\begin{aligned} Q_{i+1} &\leftarrow \alpha\mathcal{T}_*[Q_i] + (1-\alpha)Q_i + \text{noise} \\ &= Q_i - \alpha\left(Q_i - \mathcal{T}_*[Q_i]\right) + \text{noise} \\ &= Q_i - \alpha\left(\frac{1}{2}\nabla_Q \|Q - \mathcal{T}_*[Q_i]\|^2\Big|_{Q=Q_i} + \text{noise}\right) \end{aligned}$$

- Behaves like stochastic gradient descent for $f(Q) := \frac{1}{2}\|\mathcal{T}_*[Q] - Q\|^2$ but the objective is changing, i.e., $\mathcal{T}_*[Q_i]$ is a moving target
- **Stochastic Approximation Theory**: a "partial update" to ensure contraction + appropriate step size α implies convergence to the contraction fixed point: $\lim_{i\to\infty} Q_i = Q^*$
- T. Jaakkola, M. Jordan, S. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," Neural computation, 1994

Least-squares Partial Backup Version

Algorithm 7 Batch Gradient Least-squares Q-Value Iteration

1: **Init**: $Q_0(x, u)$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}$
2: **loop**
3: $\pi \leftarrow \epsilon$-greedy policy derived from Q
4: Generate episodes $\{\rho^{(k)}\}_{k=1}^{K}$ from π
5: $Q_{i+1} \leftarrow Q_i - \frac{\alpha}{2}\nabla_Q \left[\sum_{k=1}^{K}\sum_{t=0}^{T} \|\mathcal{T}_*[Q_i](x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) - Q(x_t^{(k)}, u_t^{(k)})\|^2\right]\Bigg|_{Q=Q_i}$

- Watkins Q-learning is a special case with $T = 1$