Intro Reinforcement Learning

Lecture 16
What will you take home today?

---

Reinforcement Learning

- Recap of Policy Iteration
- Sampling-based Reinforcement Learning
- Q-Learning
- Example Applications
- Policy Gradients
The Reinforcement Learning Problem

From ‘Optimal and Learning Control’ by Buchli et al.
Markov Decision Process

Markov Property

\[ \Pr \left( x_{n+1} \mid x_n, x_{n-1}, \ldots, x_0 \right) = \Pr \left( x_{n+1} \mid x_n \right) \]

\[ P^u_{x x'} = \Pr \left[ x_{n+1} = x' \mid x_n = x, u_n = u \right] \]

\[ R^u_{x x'} = E \left[ r_n \mid x_{n+1} = x', u_n = u, x_n = x \right] \]

\[ R^u_x = \sum_{x'} P^u_{x x'} R^u_{x x'} \]
The Reinforcement Learning Problem

\[ R^*_0 = r_0 + \alpha r_1 + \alpha^2 r_2 + \ldots + \alpha^n r_n + \ldots = \sum_{k=0}^{\infty} \alpha^k r_k \]

\[ \pi^* = \arg\max_\pi \mathbb{E} (R^*_0) \]

\[ R_n = \sum_{k=0}^{\infty} \alpha^k r_{n+k} \]

→ stochastic reward function
→ stochastic transition dynamics
→ stochastic policy
Deriving the Value function and Bellman Equation

Goal: Find the policy that maximizes $E \{ \text{accumulated reward} \}$

$$V^\pi(x) = E\{R_n \mid x_n = x\} = E \left\{ \sum_{k=0}^{\infty} \alpha^k r_{n+k} \mid x_n = x \right\}$$

$how$ effective a policy $is$

$$V^\pi(x) = E_{u_n, r_n, x_{n+1}} \left[ r_{n+1} + \alpha V^\pi(x) \mid x_n = x \right]$$

$= recursive$ expression $of$ Value function
Solving the Value function

\[ V^\pi(x) = \underbrace{E_{u_n,r_n,x_{n+1}} \{ r_n + \alpha V^\pi(x') \mid x_n = x \}}_{\text{Previous slide}} \]

\[
= E_{u_n} \{ E_{r_n,x_{n+1}} \{ r_n + \alpha V^\pi(x') \mid u_n = u, x_n = x \} \mid x_n = x \}
\]

Plug in known transition probabilities & reward function

\[ V^\pi(x) = \sum_u \sum_{x'} P^u_{x,x'} [R^u_{x,x'} + \alpha V^\pi(x')] \]
The Q-Function

\[ V^\pi(x) = E\{R_n \mid x_n = x\} = E \left\{ \sum_{k=0}^{\infty} \alpha^k r_{n+k} \mid x_n = x \right\} \]

\[ Q^\pi(x, u) = \text{expected accumulated reward when starting at } x, \text{ choosing } u \text{ and then following } \pi \]

\[ = E_{\pi} \left[ R_n \mid x_n = x, u_n = u \right] = E \left\{ \sum_{l=0}^{\infty} c^l r_{n+l} \mid x_n = x, u_n = u \right\} \]

\[ V^\pi(x) = \sum_{u} \pi(x, u) Q^\pi(x, u) \]
Deriving the Bellman Equation in terms of the Q-function

\[
Q^\pi(x, u) = E\{R_n \mid x_n = x, u_n = u\} = E \left\{ \sum_{k=0}^{\infty} \alpha^k r_{n+k} \mid x_n = x, u_n = u \right\}
\]

\[
Q^\pi(x_n, u) = \bar{E} \sum_{r_n, x_{n+1}} r_n + \alpha \sum_{u'} \Pi(x', u') Q^\pi(x', u')
\]

recursive expr.

\[
Q^\pi(x_n, u) = \sum_{x'} Q^\pi_{x, x'} \left[ R^u_{x, x'} + \alpha \sum_{u'} \Pi(x', u') Q^\pi(x', u') \right]
\]
Optimal policy

\[ V^*(x) \geq V^\pi(x) \]

\[ V^*(x) = \max_\pi V^\pi(x) \]

\[ Q^*(x, u) = \max_\pi Q^\pi(x, u) \]

\[ V^*(x) = \sum_a \pi^*(x, u) Q^*(x, u) \]
Policy Evaluation – How to compute $V$ and $Q$?

$$V^\pi(x) = \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \alpha V^\pi(x')]$$

$$V = AV + B$$

$$[A_{i,j}] = \alpha \sum_u \pi(x_i, u) P_{xix_j}^u$$

$$[B_i] = \sum_u \pi(x_i, u) \sum_{x'} P_{xix'}^u R_{xix'}^u$$

$$V_{k+1}(x) = \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \alpha V_k(x')]$$
Algorithm for Policy Evaluation

Algorithm 1 Iterative Policy Evaluation Algorithm

Input: \( \pi \), the policy to be evaluated

\[ V(x) = 0 \text{ for all } x \in \mathcal{X}^+ \]

repeat
\[ \Delta \leftarrow 0 \]
for each \( x \in \mathcal{X} \)
\[ u \leftarrow V(x) \]
\[ V(x) \leftarrow \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V(x')] \]
\[ \Delta \leftarrow \max(\Delta, |u - V(x)|) \]
until \( \Delta < \theta \) (a small positive number)

Return: \( V \approx V^\pi \)
Policy Evaluation with Q-Function

\[ V_{k+1}(x) = \sum_u \pi(x, u) \sum_{x'} P^u_{xx'} [R^u_{xx'} + \alpha V_k(x')] \]

\[ Q_{k+1}(x, u) = \sum_{x'} P^u_{xx'} \left[ R^u_{xx'} + \alpha \sum_{u'} \pi^*(x', u') Q_k(x', u') \right] \]
Policy Improvement

Conditions for a superior policy:

\[ \forall x \in X : \quad V^\pi'(x) \geq V^\pi(x) \]

\[ \exists x \in X : \quad V^\pi'(x) > V^\pi(x) \]

Intuition:

Given \( T^\pi \) & \( V^\pi \), how the reward changes if you use a new \( u \) but then follow the old \( T^\pi \).

Compare \( V(x) \) with \( Q(x, u) \).
Policy Improvement Theorem

\[ Q^\pi(x, \mu'(x)) \geq V^\pi(x) \]

\[ V^\pi(x) \leq V^{\pi'}(x) \]

\[ \pi'(x) = \arg\max_u Q^\pi(x, u) \]

\[ = \arg\max_u E \{ r_n + \alpha V^\pi(x_{n+1}) | x_n = x, u_n = u \} \]
Policy Iteration

\[ V \rightarrow V^\pi \rightarrow \text{evaluation} \rightarrow \text{greedy}(V) \rightarrow \text{improvement} \]

\[ \pi \rightarrow \pi^* \rightarrow V^* \]
Algorithm 2 Policy Iteration

1. Initialization
   select $V(x) \in \mathbb{R}$ and $\pi(x) \in U$ arbitrarily for all $x \in X$
2. Policy evaluation
   repeat
     $\Delta \leftarrow 0$
     for each: $x \in X$
       $v \leftarrow V(x)$
       $V(x) \leftarrow \sum_u \pi(x, u) \sum_{x'} \mathcal{P}^{u}_{x,x'}[\mathcal{R}^{u}_{x,x'} + \alpha V(x')]$
       $\Delta \leftarrow \max(\Delta, |v - V(x)|)$
   until $\Delta < \theta$ (a small positive number)
3. Policy Improvement
   policyIsStable $\leftarrow$ true
   for $x \in X$ do
     $b \leftarrow \pi(x)$
     $\pi(x) \leftarrow \arg\max_u \sum_{x'} \mathcal{P}^{u}_{x,x'}[\mathcal{R}^{u}_{x,x'} + \alpha V(x')]$
     if $b \neq \pi(x)$ then
       policyIsStable $\leftarrow$ false
     end if
   end for
   if policyIsStable then
     stop
   else
     go to 2
   end if
Return: a policy, $\pi$, such that: $\pi(x) = \arg\max_u \sum_{x'} \mathcal{P}^{u}_{x,x'}[\mathcal{R}^{u}_{x,x'} + \alpha V(x')]$
Value Iteration

\[ V_{k+1}(x) = \max_u \sum_{x'} P_{xx'}^u \left[ R_{xx'}^u + \alpha V_k(x') \right] \]

**Algorithm 3 Value Iteration**

**Initialization:** \( V(x) \in \mathbb{R} \) and \( \pi(x) \in U \) arbitrarily for all \( x \in X \)

repeat

\[ \Delta \leftarrow 0 \]

for \( x \in X \) do

\[ v \leftarrow V(x) \]

\[ V(x) \leftarrow \max_u \sum_{x'} P_{xx'}^u \left[ R_{xx'}^u + \alpha V(x') \right] \]

\[ \Delta \leftarrow \max(\Delta, |v - V(x)|) \]

end for

until \( \Delta < \theta \) (a small positive number)

Return: a policy, \( \pi \), such that: \( \pi(x) = \arg \max_u \sum_{x'} P_{xx'}^u \left[ R_{xx'}^u + \alpha V(x') \right] \)
Generalized Policy Iteration

Do the switch between Policy evaluation & improvement at any round

⇒ guaranteed to converge
Sample-based RL

Goal: Estimate reward transition function from samples

Approach: $\pi(x) = \text{argmax}_u Q^\pi(x,u)$

directly estimate

Algorithm 2 Policy Iteration

1. Initialization
   select $V(x) \in \mathbb{R}$ and $\pi(x) \in \mathbb{U}$ arbitrarily for all $x \in \mathcal{X}$

2. Policy evaluation
   repeat
     $\Delta \leftarrow 0$
     for each: $x \in \mathcal{X}$
       $v \leftarrow V(x)$
       $V(x) \leftarrow \sum_u \pi(x,u) \sum_{x'} P_u^u [R^u_{xx'} + \alpha V(x')]$
       $\Delta \leftarrow \max(\Delta, |v - V(x)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   $\text{policyIsStable} \leftarrow \text{true}$
   for $x \in \mathcal{X}$ do
     $b \leftarrow \pi(x)$
     $\pi(x) \leftarrow \text{argmax}_u \sum_{x'} P_u^u [R^u_{xx'} + \alpha V(x')]$
     if $b \neq \pi(x)$ then
       $\text{policyIsStable} \leftarrow \text{false}$
     end if
   end for
   if $\text{policyIsStable}$ then
     stop
   else
     go to 2
   end if

Return: a policy, $\pi$, such that: $\pi(x) = \text{argmax}_u \sum_{x'} P_u^u [R^u_{xx'} + \alpha V(x')]$
Policy Iteration with Q-function

\[ \pi^*(x) = \arg\max_{\pi} E \{ r_n + \alpha V^*(x') \mid x_n = x \} \]
\[ = \arg\max_{\pi} \sum_{x'} P_{xx'}^u [R_{xx'} + \alpha V^*(x')] \]

\[ T^*(x) \leftarrow \arg\max_u Q^{\pi^*}(x, u) \]

---

**Algorithm 4: Generalized Policy Iteration (GPI) using the action value function** \( Q^\pi(x, u) \)

1. Initialization
   \( Q^\pi(x, u) \in \mathbb{R} \)
2. Policy Evaluation (PE)
   repeat
   for select a pair \((x, u)\) with \(x \in \mathcal{X}, u \in \mathcal{U}\) do
   \[ v \leftarrow Q^\pi(x, u) \]
   \[ Q^\pi(x, u) \leftarrow \sum_{x'} P_{xx'}^u [R_{xx'} + \alpha \sum_u \pi (x', u) Q^\pi (x', u)] \]
   end for
   until "individual PE criterion satisfied"
3. Policy Improvement
   policy-stable \(\leftarrow\) true
   for \(x \in \mathcal{X}\) do
   \[ b \leftarrow \pi(x) \]
   \[ \pi(x) \leftarrow \arg\max_{\pi} \sum_{x'} P_{xx'}^u [R_{xx'} + \alpha V(x')] \]
   if \(b \neq \pi(x)\) then
   policy-stable \(\leftarrow\) false
   end if
   end for
   if (policy-stable == true) then
   stop;
   else
   go to 2.
   end if

From ‘Optimal and Learning Control’ by Buchli et al.
Policy Iteration with Q-function

Model-free Policy Evaluation:

\[
Q^\pi(x, u) = E\pi \{ R_n \mid x_n = x, u_n = u \}
\]

\[
\hat{Q}_N^\pi(x, u) \sim \text{taking the entire roll out into account}
\]

\[
\frac{1}{N} \sum_{i=1}^{N} \left( R^i_n + \alpha r^i_{n+1} + 2 r^i_{n+1} \right) \quad \text{more efficient}
\]

Algorithm 4 Generalized Policy Iteration (GPI) using the action value function \(Q^\pi(x, u)\)

1. Initialization
   \(Q^\pi(x, u) \in \mathbb{R}\)

2. Policy Evaluation (PE)
   repeat
   for select a pair \((x, u)\) with \(x \in X, \ u \in U\) do
   \[
   v \leftarrow Q^\pi(x, u)
   \]
   \[
   Q^\pi(x, u) \leftarrow \sum_{x'} P_{xx'}^{u} [R_{xx'}^u + \alpha \sum_{u'} \pi(x', u) Q^\pi(x', u)]
   \]
   end for
   until "individual PE criterion satisfied"

3. Policy Improvement
   policy-stable \(\leftarrow\) true
   for \(x \in X\) do
   \[
   b \leftarrow \pi(x)
   \]
   \[
   \pi(x) \leftarrow \arg\max_{u} \sum_{x'} P_{xx'}^{u} [R_{xx'}^u + \alpha V(x')]\]
   if \(b \neq \pi(x)\) then
   policy-stable \(\leftarrow\) false
   end if
   end for
   if (policy-stable == true) then
   stop;
   else
   go to 2.
   end if
end if

From 'Optimal and Learning Control' by Buchli et al.
Q-Learning
Combines Sample-based RL and Dynamic Programming

Goal: Estimate optimal Q-function

\[ Q^*(x_n, u_n) = E \left\{ r_n + \alpha \max_{u'} Q^*(x', u') \mid x_n = x, u_n = u \right\} \]

\[ \tilde{Q}^{i+1}(x_n, u_n) = \tilde{Q}^i(x_n, u_n) + \omega_{i+1} \left[ r_n^{i+1} + \alpha \max_{u'_n} \tilde{Q}^i(x'_n, u'_n) - \tilde{Q}^i(x_n, u_n) \right] \]

\[ \tilde{Q}_{N+1}^\pi(x, u) = \tilde{Q}_N^\pi(x, u) + \omega_{N+1} \cdot \left( R_{N+1} - \tilde{Q}_N^\pi(x, u) \right) \]

\[ \text{only take neighbouring states into account} \]

\[ \text{taking the entire tail of rollout} \]
Q-Learning Algorithm

**Algorithm 7 Q-Learning**

Initialize \( Q(x, u) \) arbitrarily

**Repeat** for each episode:

Initialize \( x \)

**repeat** (for each step of episode):

Choose \( u \) from \( x \) using policy derived from \( Q \) (e.g., \( \varepsilon - \) greedy)

Take action \( u \), observe \( r, x' \)

\[
Q(x, u) \leftarrow Q(x, u) + \omega [r + \gamma \max_{u'} Q(x', u') - Q(x, u)]
\]

\( x \leftarrow x' \) \hspace{1cm} \( 0 < \alpha < 1 \)

**until** \( x \) is terminal

From 'Optimal and Learning Control' by Buchli et al.
What is Deep RL or Deep Q-Learning?

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

Approximate optimal Q-function with a function parameterized \( \theta \).
Deep Q-Learning

Approximate Q function via to satisfy Bellman:

\[ Q^*(s, a) = \mathbb{E} \left[ r + y \max_{a'} Q^*(s', a') \mid s, a \right] \]

Forward Pass:

Loss function:

\[ l_i(\theta_i) = \mathbb{E}_{s, a \sim p(c)} \left[ (y_i - Q(s, a; \theta_i))^2 \right] \]

where

\[ y_i = \mathbb{E}_{s'} \left[ r + y \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a \right] \]

Backward Pass: Gradient update wrt. to Q-function parameters \( \theta \)

\[ \nabla_{\theta_i} l_i(\theta_i) = \mathbb{E} \left[ r + y \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right] \nabla_{\theta_i} Q(s, a; \theta_i) \]
Case Study: Playing Atari

Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state

Action: Game controls e.g. Left, Right, Up, Down

Reward: Score increase/decrease at each time step

Figures copyright Volodymyr Mnih et al., 2013. Reproduced with permission.
Q-network Architecture

$Q(s, a; \theta)$:
neural network with weights $\theta$

- FC-4 (Q-values)
- FC-256
- 32 4x4 conv, stride 2
- 16 8x8 conv, stride 4

Current state $s_t$: 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

Output:
$Q(s_t, a_1)$
$Q(s_t, a_2)$
$Q(s_t, a_3)$
$Q(s_t, a_4)$

[Mnih et al. NIPS Workshop 2013; Nature 2015]
Experience Replay

Q-learning is off-policy. But in practice, the policy affects you into particular states.

- Update some memory table of transitions as the game is played.
- Greater data efficiency.