Lecture 04: Monte Carlo Methods

Davit Ghazaryan

February 18, 2025

American University of Armenia

Table of contents

1. General idea and differences to dynamic programming
2. Basic Monte Carlo prediction
3. Basic Monte Carlo control
4. Extensions to Monte Carlo on-policy control
5. Monte Carlo off-policy prediction and control

Monte Carlo methods vs. Dynamic Programming

Dynamic programming:

- Model-based prediction and control
- Planning inside known MDPs

Monte Carlo methods vs. Dynamic Programming

Dynamic programming:

- Model-based prediction and control
- Planning inside known MDPs

Monte Carlo methods:

- Model-free prediction and control
- Estimating value functions and optimize policies in unknown MDPs

Monte Carlo methods vs. Dynamic Programming

Dynamic programming:

- Model-based prediction and control
- Planning inside known MDPs

Monte Carlo methods:

- Model-free prediction and control
- Estimating value functions and optimize policies in unknown MDPs
- But: still assuming finite MDP problems (or problems close to that)

Monte Carlo methods vs. Dynamic Programming

Dynamic programming:

- Model-based prediction and control
- Planning inside known MDPs

Monte Carlo methods:

- Model-free prediction and control
- Estimating value functions and optimize policies in unknown MDPs
- But: still assuming finite MDP problems (or problems close to that)
- In general: broad class of computational algorithms relying on repeated random sampling to obtain numerical results

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is model-free: no knowledge of MDP transitions / rewards
- MC learns from complete episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to episodic MDPs
    - All episodes must terminate

General Monte Carlo (MC) methods' characteristics

- Learning from experience, i.e., sequences of samples $\langle s, a, R_{t+1} \rangle$

Fig. 4.1: Monte Carlo port

General Monte Carlo (MC) methods' characteristics

- Learning from experience, i.e., sequences of samples $\langle s, a, R_{t+1} \rangle$
- Main concept: Estimation by averaging sample returns

Fig. 4.1: Monte Carlo port

General Monte Carlo (MC) methods' characteristics

- Learning from experience, i.e., sequences of samples $\langle s, a, R_{t+1} \rangle$
- Main concept: Estimation by averaging sample returns
- To guarantee well-defined returns: limited to episodic tasks

Fig. 4.1: Monte Carlo port

General Monte Carlo (MC) methods' characteristics

- Learning from experience, i.e., sequences of samples $\langle s, a, R_{t+1} \rangle$
- Main concept: Estimation by averaging sample returns
- To guarantee well-defined returns: limited to episodic tasks
- Consequence: Estimation and policy updates are only possible in an episode-by-episode way compared to step-by-step (online)

Fig. 4.1: Monte Carlo port

Example

- Scene from the movie "Next"

Table of contents

1 General idea and differences to dynamic programming

2 Basic Monte Carlo prediction

3 Basic Monte Carlo control

4 Extensions to Monte Carlo on-policy control

5 Monte Carlo off-policy prediction and control

Task description and basic solution

MC prediction problem statement

- Estimate state value $v_\pi(s)$ for a given policy π.
- Available are samples $\langle s_{t,j}, a_{t,j}, R_{t+1,j} \rangle$ for episodes $j = 1, \ldots, J$.

Task description and basic solution

MC prediction problem statement

- Estimate state value $v_\pi(s)$ for a given policy π.
- Available are samples $\langle s_{t,j}, a_{t,j}, R_{t+1,j} \rangle$ for episodes $j = 1, \ldots, J$.

MC solution approach:

- Average returns after visiting state s over episodes $j = 1, \ldots$

$$v_\pi(s) \approx \hat{v}_\pi(s) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j} = \frac{1}{J} \sum_{j=1}^{J} \sum_{i=0}^{T_j} \gamma^i R_{t+i+1,j} \,. \tag{4.1}$$

- Above, T_j denotes the terminating time step of each episode j.

Task description and basic solution

MC prediction problem statement

- Estimate state value $v_\pi(s)$ for a given policy π.
- Available are samples $\langle s_{t,j}, a_{t,j}, R_{t+1,j} \rangle$ for episodes $j = 1, \ldots, J$.

MC solution approach:

- Average returns after visiting state s over episodes $j = 1, \ldots$

$$v_\pi(s) \approx \hat{v}_\pi(s) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j} = \frac{1}{J} \sum_{j=1}^{J} \sum_{i=0}^{T_j} \gamma^i R_{t+i+1,j} \, . \tag{4.1}$$

- Above, T_j denotes the terminating time step of each episode j.
- First-visit MC: Apply (4.1) only to the first state visit per episode.

Task description and basic solution

MC prediction problem statement

- Estimate state value $v_\pi(s)$ for a given policy π.
- Available are samples $\langle s_{t,j}, a_{t,j}, R_{t+1,j} \rangle$ for episodes $j = 1, \ldots, J$.

MC solution approach:

- Average returns after visiting state s over episodes $j = 1, \ldots$

$$v_\pi(s) \approx \hat{v}_\pi(s) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j} = \frac{1}{J} \sum_{j=1}^{J} \sum_{i=0}^{T_j} \gamma^i R_{t+i+1,j} \,. \tag{4.1}$$

- Above, T_j denotes the terminating time step of each episode j.
- First-visit MC: Apply (4.1) only to the first state visit per episode.
- Every-visit MC: Apply (4.1) each time visiting a certain state per episode (if a state is visited more than one time per episode).

First-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $\hat{v}_\pi = S(s)/N(s)$
- By law of large numbers, $\hat{v}_\pi \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $\hat{v}_\pi = S(s)/N(s)$
- By law of large numbers, $\hat{v}_\pi \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Algorithmic implementation: MC-based prediction

input: a policy π to be evaluated
output: estimate of $\boldsymbol{v}^{\pi}_{\mathcal{S}}$ (i.e., value estimate for all states $s \in \mathcal{S}$)
init: $\hat{v}(s) \, \forall \, s \in \mathcal{S}$ arbitrary except $v_0(s) = 0$ if s is terminal
    $l(s) \leftarrow$ an empty list for every $s \in \mathcal{S}$
for $j = 1, \ldots, J$ *episodes* **do**
    Generate an episode following π: $s_0, a_0, R_1, \ldots, s_{T_j}, a_{T_j}, R_{T_j+1}$;
    Set $G \leftarrow 0$;
    for $t = T_j - 1, T_j - 2, T_j - 3, \ldots, 0$ *time steps* **do**
        $G \leftarrow \gamma G + R_{t+1}$;
        if $s_t \notin \langle s_0, s_1, \ldots, s_{t-1} \rangle$ **then**
            Append G to list $l(s_t)$;
            $\hat{v}(s_t) \leftarrow$ average($l(s_t)$);

Algo. 4.1: MC state-value prediction (first visit)

Incremental implementation

- Algo. 4.1 is inefficient due to large memory demand.
- Better: use incremental / recursive implementation.

Incremental implementation

- Algo. 4.1 is inefficient due to large memory demand.
- Better: use incremental / recursive implementation.
- The sample mean $\mu_1, \mu_2, \ldots$ of an arbitrary sequence $G_1, G_2, \ldots$ is:

$$\begin{aligned} \mu_J &= \frac{1}{J}\sum_{i=1}^{J} G_i = \frac{1}{J}\left[G_J + \sum_{i=1}^{J-1} G_i\right] \\ &= \frac{1}{J}\left[G_J + (J-1)\mu_{J-1}\right] = \mu_{J-1} + \frac{1}{J}\left[G_J - \mu_{J-1}\right]. \end{aligned} \tag{4.2}$$

Incremental implementation

- Algo. 4.1 is inefficient due to large memory demand.
- Better: use incremental / recursive implementation.
- The sample mean $\mu_1, \mu_2, \ldots$ of an arbitrary sequence $G_1, G_2, \ldots$ is:

$$\begin{aligned} \mu_J &= \frac{1}{J}\sum_{i=1}^{J} G_i = \frac{1}{J}\left[G_J + \sum_{i=1}^{J-1} G_i\right] \\ &= \frac{1}{J}\left[G_J + (J-1)\mu_{J-1}\right] = \mu_{J-1} + \frac{1}{J}\left[G_J - \mu_{J-1}\right]. \end{aligned} \tag{4.2}$$

- If a given decision problem is non-stationary, using a forgetting factor $\alpha \in \{\mathbb{R}|0 < \alpha < 1\}$ allows for dynamic adaption:

$$\mu_J = \mu_{J-1} + \alpha\left[G_J - \mu_{J-1}\right]. \tag{4.3}$$

Statistical properties of MC-based prediction (1)

First-time visit MC:

- Each return sample G_J is independent from the others since they were drawn from separate episodes.
- One receives i.i.d. data to estimate $\mathbb{E}\left[\hat{v}_\pi\right]$ and consequently this is bias-free.
- The estimate's variance $\mathrm{Var}\left[\hat{v}_\pi\right]$ drops with $1/n$ (n: available samples).

Statistical properties of MC-based prediction (1)

First-time visit MC:

- Each return sample G_J is independent from the others since they were drawn from separate episodes.
- One receives i.i.d. data to estimate $\mathbb{E}\left[\hat{v}_\pi\right]$ and consequently this is bias-free.
- The estimate's variance $\mathrm{Var}\left[\hat{v}_\pi\right]$ drops with $1/n$ (n: available samples).

Every-time visit MC:

- Each return sample G_J is not independent from the others since they might be obtained from same episodes.
- One receives non-i.i.d. data to estimate $\mathbb{E}\left[\hat{v}_\pi\right]$ and consequently this is biased for any $n < \infty$.
- Only in the limit $n \to \infty$ one receives $(v_\pi(s) - \mathbb{E}\left[\hat{v}_\pi(s)\right]) \to 0$.

Statistical properties of MC-based prediction (2)

- State-value estimates for each state are independent.
- One estimate does not rely on the estimate of other states (no bootstrapping compared to DP).
- Makes MC particularly attractive when one requires state-value knowledge of only one or few states.
  - Hence, generating episodes starting from the state of interest.

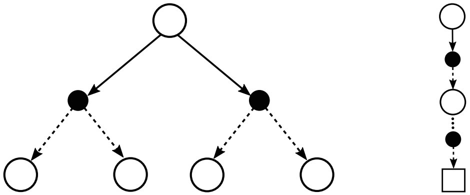

Fig. 4.2: Back-up diagrams for DP (left) and MC (right) prediction: shallow one-step back-ups with bootstrapping vs. deep back-ups over full epsiodes

Let's reuse the forest tree MDP example with *fifty-fifty policy* and discount factor $\gamma = 0.8$ plus disaster probability $\alpha = 0.2$:

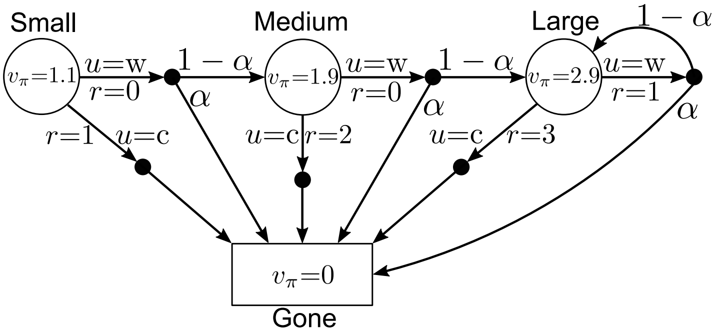

Fig. 4.3: Forest MDP with fifty-fifty-policy including state values

MC-based prediction example: forest tree MDP (2)

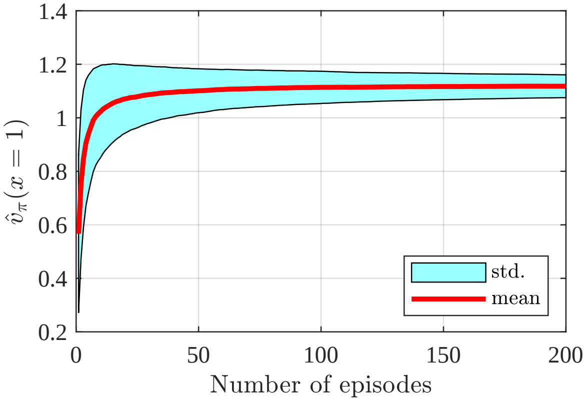

Fig. 4.4: State-value estimate of forest tree MDP initial state using MC-based prediction over the number of episodes being evaluated (mean and standard deviation are calculated based on 2000 independent runs)

MC estimation of action values

Is a model available (i.e., tuple $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$)?

- Yes: state values plus one-step prediction deliver optimal policy.
- No: action values are very useful to directly obtain optimal choices.
- Recap policy improvement from last lecture.

MC estimation of action values

Is a model available (i.e., tuple $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$)?

- Yes: state values plus one-step prediction deliver optimal policy.
- No: action values are very useful to directly obtain optimal choices.
- Recap policy improvement from last lecture.

Estimating $q_\pi(s, a)$ using MC approach:

- Analog to state values summarized in Algo. 4.1.
- Only small extension: a visit refers to a state-action pair (s, a).
- First-visit and every-visit variants exist.

MC estimation of action values

Is a model available (i.e., tuple $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$)?

- Yes: state values plus one-step prediction deliver optimal policy.
- No: action values are very useful to directly obtain optimal choices.
- Recap policy improvement from last lecture.

Estimating $q_\pi(s, a)$ using MC approach:

- Analog to state values summarized in Algo. 4.1.
- Only small extension: a visit refers to a state-action pair (s, a).
- First-visit and every-visit variants exist.

Possible problem when following a deterministic policy π:

- Certain state-action pairs (s, a) are never visited.
- Missing degree of exploration.

MC estimation of action values

Is a model available (i.e., tuple $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$)?

- Yes: state values plus one-step prediction deliver optimal policy.
- No: action values are very useful to directly obtain optimal choices.
- Recap policy improvement from last lecture.

Estimating $q_\pi(s, a)$ using MC approach:

- Analog to state values summarized in Algo. 4.1.
- Only small extension: a visit refers to a state-action pair (s, a).
- First-visit and every-visit variants exist.

Possible problem when following a deterministic policy π:

- Certain state-action pairs (s, a) are never visited.
- Missing degree of exploration.
- Workaround: exploring starts, i.e., starting episodes in random state-action pairs (s, a) and thereafter following π.

Table of contents

1. General idea and differences to dynamic programming
2. Basic Monte Carlo prediction
3. Basic Monte Carlo control
4. Extensions to Monte Carlo on-policy control
5. Monte Carlo off-policy prediction and control

Applying generalized policy iteration (GPI) to MC control

GPI concept is directly applied to MC framework using action values:

$$\pi_0 \rightarrow \hat{q}_{\pi_0} \rightarrow \pi_1 \rightarrow \hat{q}_{\pi_1} \rightarrow \cdots \pi^* \rightarrow \hat{q}_{\pi^*} \, . \tag{4.4}$$

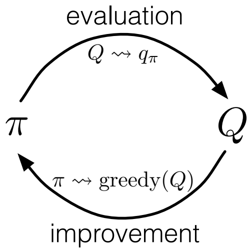

Fig. 4.5: Transferring GPI to MC-based control (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Applying generalized policy iteration (GPI) to MC control

GPI concept is directly applied to MC framework using action values:

$$\pi_0 \rightarrow \hat{q}_{\pi_0} \rightarrow \pi_1 \rightarrow \hat{q}_{\pi_1} \rightarrow \cdots \pi^* \rightarrow \hat{q}_{\pi^*}\,. \tag{4.4}$$

- Degree of freedom: Choose number of episodes to approximate $\hat{q}_{\pi_i}$.

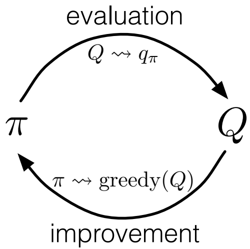

Fig. 4.5: Transferring GPI to MC-based control (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Applying generalized policy iteration (GPI) to MC control

GPI concept is directly applied to MC framework using action values:

$$\pi_0 \rightarrow \hat{q}_{\pi_0} \rightarrow \pi_1 \rightarrow \hat{q}_{\pi_1} \rightarrow \cdots \pi^* \rightarrow \hat{q}_{\pi^*} \, . \tag{4.4}$$

- Degree of freedom: Choose number of episodes to approximate $\hat{q}_{\pi_i}$.
- Policy improvement is done by greedy choices:

$$\pi(s) = \arg\max_a q(s, a) \quad \forall s \in \mathcal{S} \, . \tag{4.5}$$

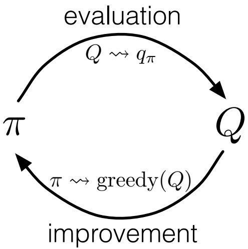

Fig. 4.5: Transferring GPI to MC-based control (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Policy improvement theorem

Assuming that one is operating in an unknown MDP, the policy improvement theorem is still valid for MC-based control:

Policy improvement for MC-based control

$$
\begin{aligned}
q_{\pi_i}(s, \pi_{i+1}(s)) &= q_{\pi_i}(s, \arg\max_a q_{\pi_i}(s, a)), \\
&= \max_a q_{\pi_i}(s, a), \\
&\geq q_{\pi_i}(s, \pi_i(s)), \\
&\geq v_{\pi_i}(s).
\end{aligned}
\tag{4.6}
$$

- Each π_{i+1} is uniformly better or just as good (if optimal) as π_i.
- Assumption: All state-action pairs are evaluated due to sufficient exploration.
  - For example using exploring starts.

Algorithmic implementation: MC-based control

output: Optimal deterministic policy π^*
init: $\pi_{i=0}(s) \in \mathcal{A}$ arbitrarily $\forall s \in \mathcal{S}$
$\hat{q}(s,a)$ arbitrarily $\forall \{s \in \mathcal{S}, a \in \mathcal{A}\}$
$n(s,a) \leftarrow$ an empty list for state-action visits $\forall \{s \in \mathcal{S}, a \in \mathcal{A}\}$
repeat
  $i \leftarrow i + 1$;
  Choose $\{s_0, a_0\}$ randomly such that all pairs have probability > 0 ;
  Generate an episode from $\{s_0, a_0\}$ following π_i until termination step T_i;
  Set $G \leftarrow 0$;
  for $k = T_i - 1, T_i - 2, T_i - 3, \ldots, 0$ *time steps* **do**
    $G \leftarrow \gamma G + R_{t+1}$;
    if $\{s_t, a_t\} \notin \langle \{s_0, a_0\}, \ldots, \{s_{t-1}, a_{t-1}\} \rangle$ **then**
      $n(s_t, a_t) \leftarrow n(s_t, a_t) + 1$;
      $\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + 1/n(s_t, a_t) \cdot (G - \hat{q}(s_t, a_t))$;
      $\pi_i(s_t) \leftarrow \arg\max_a \; \hat{q}(s_t, a)$;
until $\pi_{i+1} = \pi_i$;

Algo. 4.2: MC-based control using exploring starts (first visit)

Table of contents

1. General idea and differences to dynamic programming
2. Basic Monte Carlo prediction
3. Basic Monte Carlo control
4. Extensions to Monte Carlo on-policy control
5. Monte Carlo off-policy prediction and control

Off- and on-policy learning

- On-policy learning
  - Evaluate or improve the policy used to make decisions.
  - Agent picks actions based on its own policy.
  - Exploring starts (ES) is an on-policy method example.
  - However: ES is a restrictive assumption and not always applicable (in some cases the starting state-action pair cannot be choosen freely).

Off- and on-policy learning

- On-policy learning
  - Evaluate or improve the policy used to make decisions.
  - Agent picks actions based on its own policy.
  - Exploring starts (ES) is an on-policy method example.
  - However: ES is a restrictive assumption and not always applicable (in some cases the starting state-action pair cannot be choosen freely).
- Off-policy learning
  - Evaluate or improve a policy different from that used to generate data.
  - Agent cannot apply own actions.
  - Will be focused in the next sections.

ε-greedy as an on-policy alternative

- Exploration requirement:
  - Visit all state-action pairs with probability:

$$\pi(a|s) > 0 \quad \forall \left\{ s \in \mathcal{S}, a \in \mathcal{A} \right\} . \tag{4.7}$$

  - Policies with this characteristic are called: soft.
  - Level of exploration can be tuned during the learning process.

ε-greedy as an on-policy alternative

- Exploration requirement:
  - Visit all state-action pairs with probability:

$$\pi(a|s) > 0 \quad \forall \{s \in \mathcal{S}, a \in \mathcal{A}\} . \tag{4.7}$$

  - Policies with this characteristic are called: soft.
  - Level of exploration can be tuned during the learning process.

- ε-greedy on-policy learning
  - With probability ε the agent's choice (i.e., the policy output) is overwritten with a random action.
  - Probability of all non-greedy actions:

$$\varepsilon/|\mathcal{A}| . \tag{4.8}$$

  - Probability of the greedy action:

$$1 - \varepsilon + \varepsilon/|\mathcal{A}| . \tag{4.9}$$

  - Above, $|\mathcal{A}|$ is the size of the action space.

Algorithmic implementation ε-greedy MC-control

output: Optimal ε-greedy policy $\pi^*(a|s)$, **parameter:** $\varepsilon \in \{\mathbb{R}|0 < \varepsilon << 1\}$
init: $\pi_{i=0}(a|s)$ arbitrarily soft $\forall \{s \in \mathcal{S}, a \in \mathcal{A}\}$
$\hat{q}(s, a)$ arbitrarily $\forall \{s \in \mathcal{S}, a \in \mathcal{A}\}$
$n(s, a) \leftarrow$ an empty list counting state-action visits $\forall \{s \in \mathcal{S}, a \in \mathcal{A}\}$
repeat
Generate an episode following π_i: $s_0, a_0, R_1, \ldots, s_{T_j}, a_{T_j}, R_{T_{j+1}}$;
$i \leftarrow i + 1$;
Set $G \leftarrow 0$;
for $t = T_i - 1, T_i - 2, T_i - 3, \ldots, 0$ *time steps* **do**
$G \leftarrow \gamma G + R_{t+1}$;
if $\{s_t, a_t\} \notin \langle\{s_0, a_0\}, \ldots, \{s_{t-1}, a_{t-1}\}\rangle$ **then**
$n(s_t, a_t) \leftarrow n(s_t, a_t) + 1$;
$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + 1/n(s_t, a_t) \cdot (G - \hat{q}(s_t, a_t))$;
$\tilde{a} \leftarrow \arg\max_a \ \hat{q}(s_t, a)$;

$$\pi_i(a|s_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}|, & a = \tilde{a} \\ \varepsilon/|\mathcal{A}|, & a \neq \tilde{a} \end{cases} ;$$

until $\pi_{i+1} = \pi_i$;

Algo. 4.3: MC-based control using ε-greedy approach

ε-greedy policy improvement (1)

Theorem 4.1: Policy improvement for ε-greedy policy

Given an MDP, for any ε-greedy policy π the ε-greedy policy π' with respect to q_π is an improvement, i.e., $v_{\pi'} > v_\pi \quad \forall s \in \mathcal{S}$.

ε-greedy policy improvement (1)

Theorem 4.1: Policy improvement for ε-greedy policy

Given an MDP, for any ε-greedy policy π the ε-greedy policy π' with respect to q_π is an improvement, i.e., $v_{\pi'} > v_\pi \quad \forall s \in \mathcal{S}$.

Small proof:

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s,a), \\ &= \frac{\varepsilon}{|\mathcal{A}|} \sum_a q_\pi(s,a) + (1-\varepsilon) \max_a q_\pi(s,a), \\ &\geq \frac{\varepsilon}{|\mathcal{A}|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} q_\pi(s,a). \end{aligned} \tag{4.10}$$

In the inequality line, the second term is the weighted sum over action values given an ε-greedy policy. This weighted sum will always be smaller than or equal to $\max_a q_\pi(s,a)$.

ε-greedy policy improvement (2)

Continuation:

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &\geq \frac{\varepsilon}{|\mathcal{A}|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} q_\pi(s,a), \\
&= \frac{\varepsilon}{|\mathcal{A}|} \sum_a (q_\pi(s,a) - q_\pi(s,a)) + \sum_a \pi(a|s) q_\pi(s,a), \\
&= \sum_a \pi(a|s) q_\pi(s,a), \\
&= v_\pi(s).
\end{aligned}
\tag{4.11}
$$

ε-greedy policy improvement (2)

Continuation:

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &\geq \frac{\varepsilon}{|\mathcal{A}|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} q_\pi(s,a), \\
&= \frac{\varepsilon}{|\mathcal{A}|} \sum_a \left(q_\pi(s,a) - q_\pi(s,a)\right) + \sum_a \pi(a|s) q_\pi(s,a), \\
&= \sum_a \pi(a|s) q_\pi(s,a), \\
&= v_\pi(s).
\end{aligned}
\tag{4.11}
$$

- Policy improvement theorem is still valid when comparing ε-greedy policies against each other.
- But: There might be a non-ε-greedy policy which is better.

MC-based control example: forest tree MDP (1)

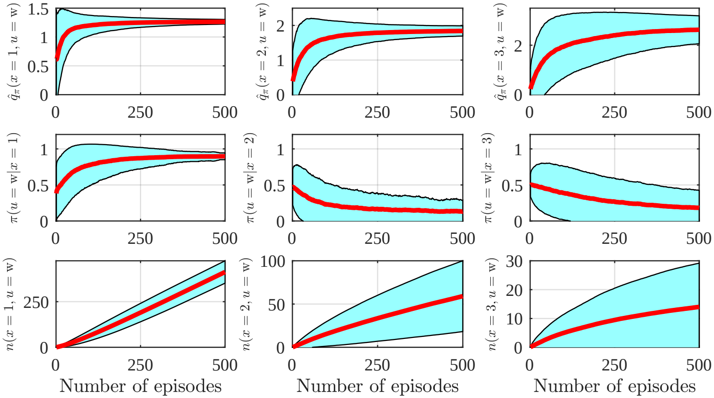

Fig. 4.6: Different estimates of forest tree MDP ($\alpha = 0.2, \gamma = 0.8$) using MC control with $\varepsilon = 0.2$ over the number of episodes. Mean is red and the standard deviation is light blue, both calculated based on 2000 independent runs.

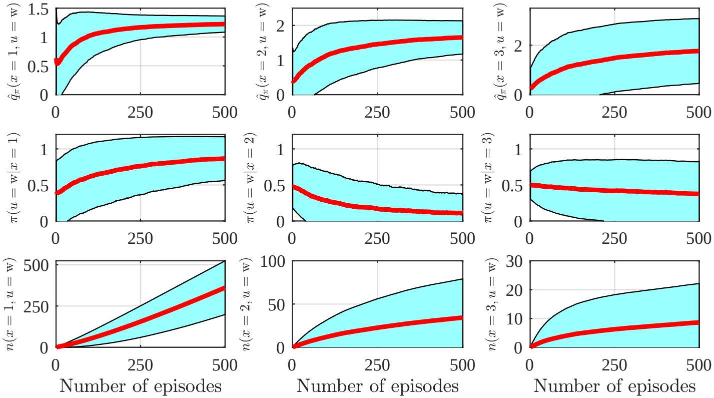

Fig. 4.7: Different estimates of forest tree MDP ($\alpha = 0.2, \gamma = 0.8$) using MC control with $\varepsilon = 0.05$ over the number of episodes. Mean is red and the standard deviation is light blue, both calculated based on 2000 independent runs.

MC-based control example: forest tree MDP (3)

Observations on forest tree MDP with ε-greedy MC-based control:

- Rather slow convergence rate: quite a number of episodes is required.

MC-based control example: forest tree MDP (3)

Observations on forest tree MDP with ε-greedy MC-based control:

- Rather slow convergence rate: quite a number of episodes is required.
- Significant uncertainty present in a single sequence.

MC-based control example: forest tree MDP (3)

Observations on forest tree MDP with ε-greedy MC-based control:

- Rather slow convergence rate: quite a number of episodes is required.
- Significant uncertainty present in a single sequence.
- Later states are less often visited and, therefore, more uncertain.

MC-based control example: forest tree MDP (3)

Observations on forest tree MDP with ε-greedy MC-based control:

- Rather slow convergence rate: quite a number of episodes is required.
- Significant uncertainty present in a single sequence.
- Later states are less often visited and, therefore, more uncertain.
- Exploration is controlled by ε: in a totally greedy policy the state $s = 3$ is not visited at all With ε-greedy this state is visited occasionally.

MC-based control example: forest tree MDP (3)

Observations on forest tree MDP with ε-greedy MC-based control:

- Rather slow convergence rate: quite a number of episodes is required.
- Significant uncertainty present in a single sequence.
- Later states are less often visited and, therefore, more uncertain.
- Exploration is controlled by ε: in a totally greedy policy the state $s = 3$ is not visited at all With ε-greedy this state is visited occasionally.
- Nevertheless, the above figures highlight that MC-based control for the forest tree MDP tend towards the optimal policies discovered by dynamic programming.

Greedy in the Limit with Infinite Exploration (GLIE)

Definition 4.1: Greedy in the limit with infinite exploration (GLIE)

A learning policy π is called GLIE if it satisfies the following two properties:

- If a state is visited infinitely often, then each action is chosen infinitely often:

$$\lim_{i\to\infty} \pi_i(a|s) = 1 \quad \forall \{s \in \mathcal{S}, a \in \mathcal{A}\} . \tag{4.12}$$

- In the limit ($i \to \infty$) the learning policy is greedy with respect to the learned action value:

$$\lim_{i\to\infty} \pi_i(a|s) = \pi(s) = \arg\max_a q(s,a) \quad \forall s \in \mathcal{S} . \tag{4.13}$$

GLIE Monte Carlo control

Theorem 4.2: Optimal decision using MC-control with ε-greedy

MC-based control using ε-greedy exploration is GLIE, if ε is decreased at rate

$$\varepsilon_i = \frac{1}{i} \tag{4.14}$$

with i being the increasing episode index. In this case,

$$\hat{q}(s, a) = q^*(s, a) \tag{4.15}$$

follows.

GLIE Monte Carlo control

Theorem 4.2: Optimal decision using MC-control with ε-greedy

MC-based control using ε-greedy exploration is GLIE, if ε is decreased at rate

$$\varepsilon_i = \frac{1}{i} \tag{4.14}$$

with i being the increasing episode index. In this case,

$$\hat{q}(s,a) = q^*(s,a) \tag{4.15}$$

follows.

Remarks:

- Limited feasibility: infinite number of episodes required.
- ε-greedy is an undirected and unmonitored random exploration strategy. Can that be the most efficient way of learning?

Table of contents

1. General idea and differences to dynamic programming
2. Basic Monte Carlo prediction
3. Basic Monte Carlo control
4. Extensions to Monte Carlo on-policy control
5. Monte Carlo off-policy prediction and control

Off-policy learning background

Drawback of on-policy learning:

- Only a compromise: comes with inherent exploration but at the cost of learning action values for a near-optimal policy.

Off-policy learning background

Drawback of on-policy learning:

- Only a compromise: comes with inherent exploration but at the cost of learning action values for a near-optimal policy.

Idea off-policy learning:

- Use two separated policies:
  - Behavior policy $b(a|s)$: explores in order to generate experience.
  - Target policy $\pi(a|s)$: learns from that experience to become the optimal policy.

Off-policy learning background

Drawback of on-policy learning:

- Only a compromise: comes with inherent exploration but at the cost of learning action values for a near-optimal policy.

Idea off-policy learning:

- Use two separated policies:
  - Behavior policy $b(a|s)$: explores in order to generate experience.
  - Target policy $\pi(a|s)$: learns from that experience to become the optimal policy.
- Use cases:
  - Learn from observing humans or other agents/controllers.

Off-policy learning background

Drawback of on-policy learning:

- Only a compromise: comes with inherent exploration but at the cost of learning action values for a near-optimal policy.

Idea off-policy learning:

- Use two separated policies:
  - Behavior policy $b(a|s)$: explores in order to generate experience.
  - Target policy $\pi(a|s)$: learns from that experience to become the optimal policy.
- Use cases:
  - Learn from observing humans or other agents/controllers.
  - Re-use experience generated from old policies $(\pi_0, \pi_1, \ldots)$.

Off-policy learning background

Drawback of on-policy learning:

- Only a compromise: comes with inherent exploration but at the cost of learning action values for a near-optimal policy.

Idea off-policy learning:

- Use two separated policies:
  - Behavior policy $b(a|s)$: explores in order to generate experience.
  - Target policy $\pi(a|s)$: learns from that experience to become the optimal policy.
- Use cases:
  - Learn from observing humans or other agents/controllers.
  - Re-use experience generated from old policies $(\pi_0, \pi_1, \ldots)$.
  - Learn about multiple policies while following one policy.

Off-policy prediction problem statement

MC off-policy prediction problem statement

- Estimate v_π and/or q_π while following $b(a|s)$.
- Both policies are considered fixed (prediction assumption).

Off-policy prediction problem statement

MC off-policy prediction problem statement

- Estimate v_π and/or q_π while following $b(a|s)$.
- Both policies are considered fixed (prediction assumption).

Requirement:

- Coverage: Every action taken under π must be (at least occasionally) taken under b, too. Hence, it follows:

$$\pi(a|s) > 0 \Rightarrow b(a|s) > 0 \quad \forall \left\{s \in \mathcal{S}, a \in \mathcal{A}\right\}. \tag{4.16}$$

Off-policy prediction problem statement

MC off-policy prediction problem statement

- Estimate v_π and/or q_π while following $b(a|s)$.
- Both policies are considered fixed (prediction assumption).

Requirement:

- Coverage: Every action taken under π must be (at least occasionally) taken under b, too. Hence, it follows:

$$\pi(a|s) > 0 \Rightarrow b(a|s) > 0 \quad \forall \left\{s \in \mathcal{S}, a \in \mathcal{A}\right\}. \tag{4.16}$$

- Consequences from that:
  - In any state b is not identical to π, b must be stochastic.
  - Nevertheless: π might be deterministic (e.g., control applications) or stochastic.

Importance sampling

Probability of observing a certain trajectory on random variables $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$ starting in S_t while following π:

$$\begin{aligned}\mathbb{P}\left[A_t, S_{t+1}, A_{t+1}, \ldots, S_T | A_t, \pi\right] &= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1})\cdots, \\ &= \prod_t^{T-1} \pi(A_t|S_t)p(S_{t+1}|S_t, A_t).\end{aligned} \tag{4.17}$$

Above p is the state-transition probability.

Probability of observing a certain trajectory on random variables $A_t, S_{t+1}, A_{t+1}, \dots, S_T$ starting in S_t while following π:

$$\begin{aligned} \mathbb{P}\left[A_t, S_{t+1}, A_{t+1}, \dots, S_T | A_t, \pi\right] &= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1})\cdots, \\ &= \prod_t^{T-1} \pi(A_t|S_t)p(S_{t+1}|S_t, A_t). \end{aligned} \tag{4.17}$$

Above p is the state-transition probability.

Definition 4.2: Importance sampling ratio

The relative probability of a trajectory under the target and behavior policy, the importance sampling ratio, from sample step k to T is:

$$\rho_{k:T} = \frac{\prod_t^{T-1} \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)}{\prod_t^{T-1} b(A_t|S_t)p(S_{t+1}|S_t, A_t)} = \frac{\prod_t^{T-1} \pi(A_t|S_t)}{\prod_t^{T-1} b(A_t|S_t)}. \tag{4.18}$$

Importance sampling for Monte Carlo prediction

Definition 4.3: State-value estimation via Monte Carlo importance sampling

Estimating the state value v_π following a behavior policy b using (ordinary) importance sampling (OIS) results in scaling and averaging the sampled returns by the importance sampling ratio per episode:

$$\hat{v}_\pi(s_t) = \frac{\sum_{t \in \mathcal{T}(s_t)} \rho_{t:T(t)} G_t}{|\mathcal{T}(s_t)|}. \tag{4.19}$$

Notation remark:

- $\mathcal{T}(s_t)$: set of all time steps in which the state s_t is visited.
- $T(t)$: Termination of a specific episode starting from t.

Importance sampling for Monte Carlo prediction

Definition 4.3: State-value estimation via Monte Carlo importance sampling

Estimating the state value v_π following a behavior policy b using (ordinary) importance sampling (OIS) results in scaling and averaging the sampled returns by the importance sampling ratio per episode:

$$\hat{v}_\pi(s_t) = \frac{\sum_{t \in \mathcal{T}(s_t)} \rho_{t:T(t)} G_t}{|\mathcal{T}(s_t)|}. \tag{4.19}$$

Notation remark:

- $\mathcal{T}(s_t)$: set of all time steps in which the state s_t is visited.
- $T(t)$: Termination of a specific episode starting from t.

General remark:

- From (4.18) it can be seen that $\hat{v}$ is bias-free (first-visit assumption).
- However, if ρ is large (distinctly different policies) the estimate's variance is large (i.e., uncertain for small numbers of samples).

Off-policy Monte Carlo control

Just put everything together:

- MC-based control utilizing GPI (cf. Fig. 4.5),
- Off-policy learning based on importance sampling (or variants like weighted importance sampling, cf. Barto/Sutton book chapter 5.5).

Off-policy Monte Carlo control

Just put everything together:

- MC-based control utilizing GPI (cf. Fig. 4.5),
- Off-policy learning based on importance sampling (or variants like weighted importance sampling, cf. Barto/Sutton book chapter 5.5).

Requirement for off-policy MC-based control:

- Coverage: behavior policy b has nonzero probability of selecting actions that might be taken by the target policy π.
- Consequence: behavior policy b is soft (e.g., ε-soft).

Summary

- MC methods allow model-free learning of value functions and optimal policies from experience in the form of sampled episodes.
- Using deep back-ups over full episodes, MC is largely based on averaging returns.

Summary

- MC methods allow model-free learning of value functions and optimal policies from experience in the form of sampled episodes.
- Using deep back-ups over full episodes, MC is largely based on averaging returns.
- MC-based control reuses generalized policy iteration (GPI), i.e., mixing policy evaluation and improvement.

Summary

- MC methods allow model-free learning of value functions and optimal policies from experience in the form of sampled episodes.
- Using deep back-ups over full episodes, MC is largely based on averaging returns.
- MC-based control reuses generalized policy iteration (GPI), i.e., mixing policy evaluation and improvement.
- Maintaining sufficient exploration is important:
  - Exploring starts: not feasible in all applications but simple.
  - On-policy ε-greedy learning: trade-off between optimality and exploration cannot be resolved easily.
  - Off-policy learning: agent learns about a (possibly deterministic) target policy from an exploratory, soft behavior policy.

Summary

- MC methods allow model-free learning of value functions and optimal policies from experience in the form of sampled episodes.
- Using deep back-ups over full episodes, MC is largely based on averaging returns.
- MC-based control reuses generalized policy iteration (GPI), i.e., mixing policy evaluation and improvement.
- Maintaining sufficient exploration is important:
  - Exploring starts: not feasible in all applications but simple.
  - On-policy ε-greedy learning: trade-off between optimality and exploration cannot be resolved easily.
  - Off-policy learning: agent learns about a (possibly deterministic) target policy from an exploratory, soft behavior policy.
- Importance sampling transforms expectations from the behavior to the target policy.
  - This estimation task comes with a bias-variance-dilemma.
  - Slow learning can result from ineffective experience usage in MC methods.