Monte Carlo Methods

(Estimators, On-policy/Off-policy Learning)

Julie Nutini

MLRG - Winter Term 2

January 24^{th}, 2017

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.
- Do not assume complete knowledge of environment.
  $\rightarrow$ Learn from experience.
- Sample sequences of states, actions and rewards.

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.
- Do not assume complete knowledge of environment.
  $\rightarrow$ Learn from experience.
- Sample sequences of states, actions and rewards.
  - On-line experience: No model necessary, attains optimality.

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.
- Do not assume complete knowledge of environment.
  - $\rightarrow$ Learn from experience.
- Sample sequences of states, actions and rewards.
  - On-line experience: No model necessary, attains optimality.
  - Simulated experience: No need for full model.
    - Sample according to desired probability distributions.

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.
- Do not assume complete knowledge of environment.

  $\rightarrow$ Learn from experience.
- Sample sequences of states, actions and rewards.
  - On-line experience: No model necessary, attains optimality.
  - Simulated experience: No need for full model.
    - Sample according to desired probability distributions.
- Solve RL problem by averaging complete sample returns.
  - Episodic tasks ensure well-defined returns are available.

Monte Carlo Methods

- Monte Carlo (MC) methods are learning methods, used for estimating value functions and discovering optimal policies.
- Do not assume complete knowledge of environment.
  - $\rightarrow$ Learn from experience.
- Sample sequences of states, actions and rewards.
  - On-line experience: No model necessary, attains optimality.
  - Simulated experience: No need for full model.
    - Sample according to desired probability distributions.
- Solve RL problem by averaging complete sample returns.
  - Episodic tasks ensure well-defined returns are available.
  - Incremental in an episode-by-episode sense.
    - Update value estimates/policies after completion of episode.

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^\pi(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^{\pi}(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.
- **Given**: Some number of episodes under π which contain s.

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^\pi(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.
- **Given**: Some number of episodes under π which contain s.
- **Idea**: Average returns observed after visits to s.

$\rightarrow$ Average converges to expected value with $\uparrow$ # returns.

  - (Underlying idea to all Monte Carlo methods.)

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^{\pi}(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.
- **Given**: Some number of episodes under π which contain s.
- **Idea**: Average returns observed after visits to s.

  $\rightarrow$ Average converges to expected value with $\uparrow$ # returns.
    - (Underlying idea to all Monte Carlo methods.)
- Each occurrence of state s in an episode is called a visit.

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^\pi(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.
- **Given**: Some number of episodes under π which contain s.
- **Idea**: Average returns observed after visits to s.

  $\rightarrow$ Average converges to expected value with $\uparrow$ # returns.
    - (Underlying idea to all Monte Carlo methods.)

- Each occurrence of state s in an episode is called a visit.
  - First-visit MC: Average returns for first time s visited in episode.
  - Every-visit MC: Average returns for every time s visited in episode.

Monte Carlo Policy Evaluation

- **Goal**: Learn state-value function $V^\pi(s)$ for given policy π.
  - Value of a state is the expected return (expected cumulative future discounted reward) starting from s.
- **Given**: Some number of episodes under π which contain s.
- **Idea**: Average returns observed after visits to s.

  $\rightarrow$ Average converges to expected value with $\uparrow$ # returns.
    - (Underlying idea to all Monte Carlo methods.)

- Each occurrence of state s in an episode is called a visit.
  - First-visit MC: Average returns for first time s visited in episode.
  - Every-visit MC: Average returns for every time s visited in episode.
- Both converge asymptotically.

First-Visit Monte Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow$ average$(Returns(s))$

First-Visit Monte Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

- Each return is an i.i.d. estimate of $V^\pi(s)$.

First-Visit Monte Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

- Each return is an i.i.d. estimate of $V^{\pi}(s)$.
- Every average is an unbiased estimate, s.d. of error falls as $1/\sqrt{n}$.

First-Visit Monte Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated
$V \leftarrow$ an arbitrary state-value function
$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π
(b) For each state s appearing in the episode:
  $R \leftarrow$ return following the first occurrence of s
  Append R to $Returns(s)$
  $V(s) \leftarrow \text{average}(Returns(s))$

- Each return is an i.i.d. estimate of $V^{\pi}(s)$.
- Every average is an unbiased estimate, s.d. of error falls as $1/\sqrt{n}$.
- Sequence of averages converges to expected value of $V^{\pi}(s)$.

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.
- **States** (200 of them):
  - Current sum (12-21).
  - Dealer's showing card (ace-10).
  - Do I have a useable ace?

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.
- **States** (200 of them):
  - Current sum (12-21).
  - Dealer's showing card (ace-10).
  - Do I have a useable ace?
- **Reward**: +1 for winning, 0 for a draw, -1 for losing.
  - All rewards within game are 0, do not discount ($\gamma = 0$).

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.
- **States** (200 of them):
  - Current sum (12-21).
  - Dealer's showing card (ace-10).
  - Do I have a useable ace?
- **Reward**: +1 for winning, 0 for a draw, -1 for losing.
  - All rewards within game are 0, do not discount ($\gamma = 0$).
- **Actions**:
  - Stick (stop receiving cards).
  - Hit (receive another card).

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.
- **States** (200 of them):
  - Current sum (12-21).
  - Dealer's showing card (ace-10).
  - Do I have a useable ace?
- **Reward**: +1 for winning, 0 for a draw, -1 for losing.
  - All rewards within game are 0, do not discount ($\gamma = 0$).
- **Actions**:
  - Stick (stop receiving cards).
  - Hit (receive another card).
- **Policy**: Stick if my sum is 20 or 21, otherwise hit.

Example: Blackjack

- **Goal**: Card sum greater than dealer without exceeding 21.
- **States** (200 of them):
  - Current sum (12-21).
  - Dealer's showing card (ace-10).
  - Do I have a useable ace?
- **Reward**: +1 for winning, 0 for a draw, -1 for losing.
  - All rewards within game are 0, do not discount ($\gamma = 0$).
- **Actions**:
  - Stick (stop receiving cards).
  - Hit (receive another card).
- **Policy**: Stick if my sum is 20 or 21, otherwise hit.

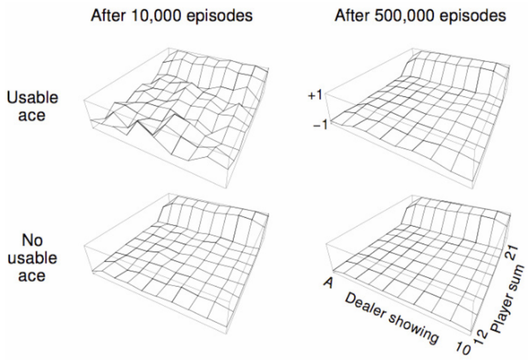
$\rightarrow$ Find state-value function for policy by MC approach.

Blackjack Value Functions

- Simulate many blackjack games using policy π.
- Average returns following each state (first-visit MC).

Blackjack Value Functions

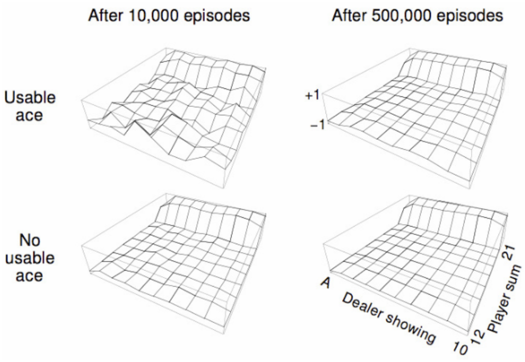
- Simulate many blackjack games using policy π.
- Average returns following each state (first-visit MC).



- Higher number of games (episodes), better approximation.

Blackjack Value Functions

- Simulate many blackjack games using policy π.
- Average returns following each state (first-visit MC).



- Higher number of games (episodes), better approximation.
- Estimates for states with useable ace less certain.

Dynamic Programming vs. Monte Carlo

- Dynamic programming (DP): full knowledge of environment.
  - e.g., blackjack, naturally formulated as episodic finite MDP

Dynamic Programming vs. Monte Carlo

- Dynamic programming (DP): full knowledge of environment.
  - e.g., blackjack, naturally formulated as episodic finite MDP
    - Player's sum is 14, chooses to stick.
    - What is expected reward as function of dealer's hand?

Dynamic Programming vs. Monte Carlo

- Dynamic programming (DP): full knowledge of environment.
  - e.g., blackjack, naturally formulated as episodic finite MDP
    - Player's sum is 14, chooses to stick.
    - What is expected reward as function of dealer's hand?
    - Requires all expected rewards and transition probabilities to be computed prior to applying DP

Dynamic Programming vs. Monte Carlo

- Dynamic programming (DP): full knowledge of environment.
  - e.g., blackjack, naturally formulated as episodic finite MDP
    - Player's sum is 14, chooses to stick.
    - What is expected reward as function of dealer's hand?
    - Requires all expected rewards and transition probabilities to be computed prior to applying DP$\rightarrow$ complex, error-prone.

Dynamic Programming vs. Monte Carlo

- Dynamic programming (DP): full knowledge of environment.
  - e.g., blackjack, naturally formulated as episodic finite MDP
    - Player's sum is 14, chooses to stick.
    - What is expected reward as function of dealer's hand?
    - Requires all expected rewards and transition probabilities to be computed prior to applying DP$\rightarrow$ complex, error-prone.

$\rightarrow$ Generating sample games easy.

- MC methods can be better, even when complete knowledge of environment's dynamics is known.

Backup Diagram for Monte Carlo

- Shows all transitions, leaf nodes from root node whose rewards and estimated values contribute to update.

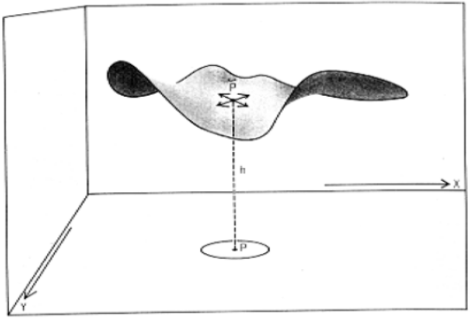
Backup Diagram for Monte Carlo

- Shows all transitions, leaf nodes from root node whose rewards and estimated values contribute to update.

- Entire episode.
    - Rather than one-step transitions.
- Only one choice at each state.
    - DP explores all possible transitions.
- MC does not bootstrap.
    - Independent estimates for each state.
- Time required to estimate one state independent of total number of states.

The Power of Monte Carlo

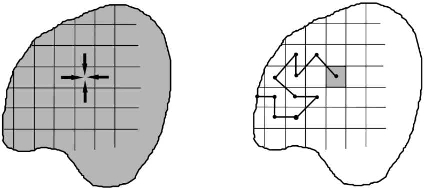
- E.g., elastic membrane (Dirichlet Problem)
  - How do we compute the shape of the surface?
    → Geometry of wire frame is known.



The Power of Monte Carlo

Relaxation

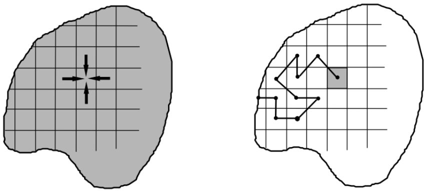
Kakutani's algorithm, 1945



1. Height at any point is average of heights in small circle around point.

The Power of Monte Carlo

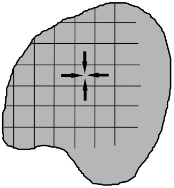
Relaxation　　　　Kakutani's algorithm, 1945



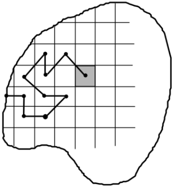
1. Height at any point is average of heights in small circle around point.
   - Solve by iterating, adjust towards average of neighbours.

The Power of Monte Carlo

Relaxation

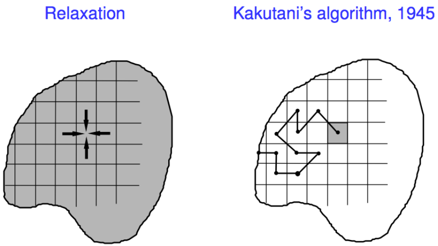


Kakutani's algorithm, 1945



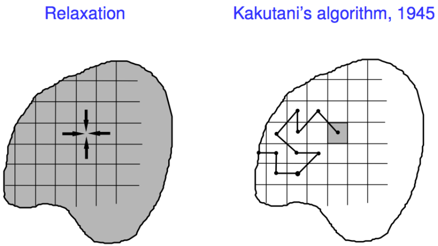
1. Height at any point is average of heights in small circle around point.
   - Solve by iterating, adjust towards average of neighbours.
2. Expected value of height at boundary approximates height of surface at starting point.

The Power of Monte Carlo



1. Height at any point is average of heights in small circle around point.
   - Solve by iterating, adjust towards average of neighbours.
2. Expected value of height at boundary approximates height of surface at starting point.
   - Take random walk until reach boundary.
   - Average boundary heights of many walks.

The Power of Monte Carlo



1. Height at any point is average of heights in small circle around point.
   - Solve by iterating, adjust towards average of neighbours.
2. Expected value of height at boundary approximates height of surface at starting point.
   - Take random walk until reach boundary.
   - Average boundary heights of many walks.

$\rightarrow$ Local consistency.

Monte Carlo Estimation of Action Values (Q)

- MC is most useful when a model is not available.
    - With model, state values are sufficient to determine policy.
        - Choose action that leads to best reward/next state.

Monte Carlo Estimation of Action Values (Q)

- MC is most useful when a model is not available.
  - With model, state values are sufficient to determine policy.
    - Choose action that leads to best reward/next state.
  - Without model, need to also estimate action values.

Monte Carlo Estimation of Action Values (Q)

- MC is most useful when a model is not available.
  - With model, state values are sufficient to determine policy.
    - Choose action that leads to best reward/next state.
  - Without model, need to also estimate action values.

    $\rightarrow$ We want to learn Q^*.

Monte Carlo Estimation of Action Values (Q)

- MC is most useful when a model is not available.
  - With model, state values are sufficient to determine policy.
    - Choose action that leads to best reward/next state.
  - Without model, need to also estimate action values.
    - $\rightarrow$ We want to learn Q^*.
- Policy evaluation problem for action values:
  - Estimate $Q^\pi(s, a)$, the expected return starting from state s, taking action a, then following policy π.

Monte Carlo Estimation of Action Values (Q)

- Average returns following first visit to s in each episode where a was selected.

Monte Carlo Estimation of Action Values (Q)

- Average returns following first visit to s in each episode where a was selected.
- Converges asymptotically *if* every state-action pair visited.

Monte Carlo Estimation of Action Values (Q)

- Average returns following first visit to s in each episode where a was selected.
- Converges asymptotically *if* every state-action pair visited.
  - Many relevant state-action pairs may never be visited.
  - E.g., π is deterministic, observe returns from only one action from each state $\rightarrow$ no returns to average.

Monte Carlo Estimation of Action Values (Q)

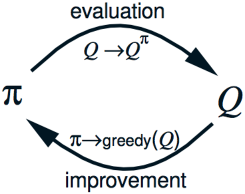
- Average returns following first visit to s in each episode where a was selected.
- Converges asymptotically *if* every state-action pair visited.
    - Many relevant state-action pairs may never be visited.
    - E.g., π is deterministic, observe returns from only one action from each state $\rightarrow$ no returns to average.
- Need to maintain exploration.
    - **Exploring starts**: Every state-action pair has non-zero probability of being starting pair.

Monte Carlo Estimation of Action Values (Q)

- Average returns following first visit to s in each episode where a was selected.
- Converges asymptotically *if* every state-action pair visited.
  - Many relevant state-action pairs may never be visited.
  - E.g., π is deterministic, observe returns from only one action from each state $\rightarrow$ no returns to average.
- Need to maintain exploration.
  - **Exploring starts**: Every state-action pair has non-zero probability of being starting pair.
  - **Alternative**: Only consider policies that are stochastic with nonzero probability of selecting all actions (later).

Monte Carlo Control

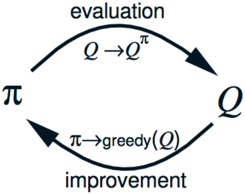
- Using MC estimation to approximate optimal policies.



$$\pi_0 \xrightarrow{\text{E}} Q^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} Q^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} Q^*$$

Monte Carlo Control

- Using MC estimation to approximate optimal policies.

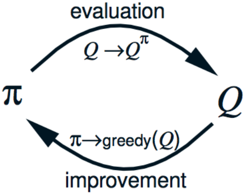


$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi^* \xrightarrow{E} Q^*$$

- Policy evaluation (E):
  - Complete policy evaluation using MC methods.

Monte Carlo Control

- Using MC estimation to approximate optimal policies.



$$\pi_0 \xrightarrow{\text{E}} Q^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} Q^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} Q^*$$

- Policy evaluation (E):
  - Complete policy evaluation using MC methods.
- Policy improvement (I):
  - Greedify policy wrt current action-value function,

$$\pi(s) = \underset{a}{\text{argmax}}\, Q(s,a).$$

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$Q^{\pi_k}(s, \pi_{k+1}(s)) \quad = \quad Q^{\pi_k}(s, \operatorname*{argmax}_a Q^{\pi_k}(s, a))$$

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \underset{a}{\operatorname{argmax}}\, Q^{\pi_k}(s,a)) \\ &= \max_a Q^{\pi_k}(s,a) \end{aligned}$$

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \operatorname*{argmax}_a Q^{\pi_k}(s, a)) \\ &= \max_a Q^{\pi_k}(s, a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \quad \text{(*corrected)} \end{aligned}$$

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$
\begin{aligned}
Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \operatorname*{argmax}_a Q^{\pi_k}(s, a)) \\
&= \max_a Q^{\pi_k}(s, a) \\
&\geq Q^{\pi_k}(s, \pi_k(s)) \quad \text{(*corrected)} \\
&= V^{\pi_k}(s).
\end{aligned}
$$

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \underset{a}{\mathsf{argmax}}\, Q^{\pi_k}(s,a)) \\ &= \max_a Q^{\pi_k}(s,a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \qquad \text{(*corrected)} \\ &= V^{\pi_k}(s). \end{aligned}$$

- By policy improvement theorem, π_{k+1} better than π_k.
- Assures convergence to optimal policy and value function.
  - $\rightarrow$ Assumes exploring starts and infinite number of episodes.

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$
\begin{aligned}
Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \operatorname*{argmax}_a Q^{\pi_k}(s,a)) \\
&= \max_a Q^{\pi_k}(s,a) \\
&\geq Q^{\pi_k}(s, \pi_k(s)) \quad \text{(*corrected)} \\
&= V^{\pi_k}(s).
\end{aligned}
$$

- By policy improvement theorem, π_{k+1} better than π_k.
- Assures convergence to optimal policy and value function.
  - $\rightarrow$ Assumes exploring starts and infinite number of episodes.
- To solve the latter:
  - Update only to a given level of performance (approx. Q^{π_k}).

Convergence of MC Control

- Greedified policy meets conditions for policy improvement:

$$
\begin{aligned}
Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \underset{a}{\text{argmax}}\, Q^{\pi_k}(s,a)) \\
&= \max_a Q^{\pi_k}(s,a) \\
&\geq Q^{\pi_k}(s, \pi_k(s)) \quad \text{(*corrected)} \\
&= V^{\pi_k}(s).
\end{aligned}
$$

- By policy improvement theorem, π_{k+1} better than π_k.
- Assures convergence to optimal policy and value function.
  - $\rightarrow$ Assumes exploring starts and infinite number of episodes.
- To solve the latter:
  - Update only to a given level of performance (approx. Q^{π_k}).
  - Alternate between evaluation & improvement per episode.

Monte Carlo with Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s, a) \leftarrow$ arbitrary
    $\pi(s) \leftarrow$ arbitrary
    $Returns(s, a) \leftarrow$ empty list

Repeat forever:
    (a) Generate an episode using exploring starts and π
    (b) For each pair s, a appearing in the episode:
        $R \leftarrow$ return following the first occurrence of s, a
        Append R to $Returns(s, a)$
        $Q(s, a) \leftarrow$ average($Returns(s, a)$)
    (c) For each s in the episode:
        $\pi(s) \leftarrow \arg\max_a Q(s, a)$

Monte Carlo with Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow$ arbitrary

$\pi(s) \leftarrow$ arbitrary

$Returns(s, a) \leftarrow$ empty list

Repeat forever:

(a) Generate an episode using exploring starts and π

(b) For each pair s, a appearing in the episode:

$R \leftarrow$ return following the first occurrence of s, a

Append R to $Returns(s, a)$

$Q(s, a) \leftarrow \text{average}(Returns(s, a))$

(c) For each s in the episode:

$\pi(s) \leftarrow \arg\max_a Q(s, a)$

- All returns averaged, irrespective of specific policy.

Monte Carlo with Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

Repeat forever:
(a) Generate an episode using exploring starts and π
(b) For each pair s, a appearing in the episode:
$R \leftarrow$ return following the first occurrence of s, a
Append R to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
(c) For each s in the episode:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

- All returns averaged, irrespective of specific policy.
- Convergence to optimal fixed point seems inevitable.
- Open problem: Proving convergence to optimal fixed point.

Example: Blackjack

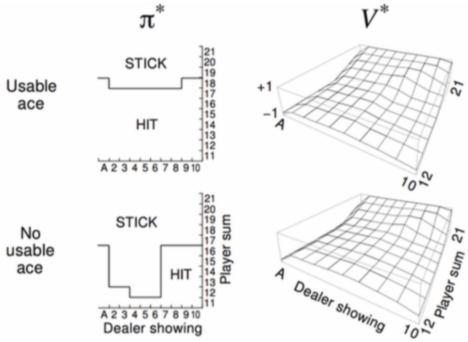
- Applying MC with exploring starts to blackjack problem.
- Use same initial policy.

Example: Blackjack

- Applying MC with exploring starts to blackjack problem.
- Use same initial policy.
- Find optimal policy and state-value function.

Example: Blackjack

- Applying MC with exploring starts to blackjack problem.
- Use same initial policy.
- Find optimal policy and state-value function.



- Randomly select with equal prob. dealer's cards, player's sum and whether or not player has usable ace.

On-Policy Monte Carlo Control

- How to avoid exploring starts?

On-Policy Monte Carlo Control

- How to avoid exploring starts?
- On-policy: Evaluate/improve policy while using for control.
  - Need soft policies: $\pi(s,a) > 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.

On-Policy Monte Carlo Control

- How to avoid exploring starts?
- On-policy: Evaluate/improve policy while using for control.
  - Need soft policies: $\pi(s,a) > 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.
  - E.g., An ϵ-greedy policy is an example of ϵ-soft policy,

$$\pi(s,a) \geq \frac{\epsilon}{|\mathcal{A}(s)|}, \quad \forall\ s, a, \text{ and some } \epsilon > 0.$$

On-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list
  $\pi \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s, a appearing in the episode:
    $R \leftarrow$ return following the first occurrence of s, a
    Append R to $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
  (c) For each s in the episode:
    $a^* \leftarrow \arg\max_a Q(s,a)$
    For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

- Encourages exploration of nongreedy actions.

Learning About π While Following π'

- Suppose episodes are generated from different policy.

Learning About π While Following π'

- Suppose episodes are generated from different policy.
- Can we learn the value function for a policy given only “off” policy experience?

Learning About π While Following π'

- Suppose episodes are generated from different policy.
- Can we learn the value function for a policy given only “off” policy experience?
  - **Yes**! Requires that $\pi(s,a) > 0$ implies $\pi'(s,a) > 0$.

Learning About π While Following π'

- Suppose episodes are generated from different policy.
- Can we learn the value function for a policy given only "off" policy experience?
  - **Yes**! Requires that $\pi(s,a) > 0$ implies $\pi'(s,a) > 0$.

- We have n_s returns, $R_i(s)$, from state s, with:
  - probability $p_i(s)$ of being generated by π
  - probability $p'_i(s)$ of being generated by π'
- Estimate using weighted importance sampling:

$$V_\pi(s) \approx \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

Learning About π While Following π'

- Suppose episodes are generated from different policy.
- Can we learn the value function for a policy given only "off" policy experience?
  - **Yes**! Requires that $\pi(s, a) > 0$ implies $\pi'(s, a) > 0$.

- We have n_s returns, $R_i(s)$, from state s, with:
  - probability $p_i(s)$ of being generated by π
  - probability $p_i'(s)$ of being generated by π'
- Estimate using weighted importance sampling:

$$V_\pi(s) \approx \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)}}$$

- Depends on the environmental probabilities $p_i(s)$ and $p_i'(s)$.
  - Normally considered unknown in MC applications.

Learning About π While Following π'

- However,

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k)\mathcal{P}_{s_k s_{k+1}}^{a_k}$$

Learning About π While Following π'

- However,

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k)\mathcal{P}_{s_k s_{k+1}}^{a_k}$$

and

$$\frac{p_i(s_t)}{p'_i(s_t)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k)\mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k)\mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}.$$

Learning About π While Following π'

- However,

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}$$

and

$$\frac{p_i(s_t)}{p_i'(s_t)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}.$$

$\rightarrow$ The weights only depend on the two policies!

Off-Policy Monte Carlo Control

- Alternative to exploring starts and on-policy.
- **On-policy**: evaluate/improve policy while using for control.
- **Off-policy**: separates these two functions.

Off-Policy Monte Carlo Control

- Alternative to exploring starts and on-policy.
- **On-policy**: evaluate/improve policy while using for control.
- **Off-policy**: separates these two functions.
  - Behaviour policy: generates behaviour in environment.
    - Continually sample actions, ϵ-soft.

Off-Policy Monte Carlo Control

- Alternative to exploring starts and on-policy.
- **On-policy**: evaluate/improve policy while using for control.
- **Off-policy**: separates these two functions.
  - Behaviour policy: generates behaviour in environment.
    - Continually sample actions, ϵ-soft.
  - Estimation policy: evaluated and improved.
    - Deterministic, greedy.

Off-Policy Monte Carlo Control

- Alternative to exploring starts and on-policy.
- **On-policy**: evaluate/improve policy while using for control.
- **Off-policy**: separates these two functions.
    - Behaviour policy: generates behaviour in environment.
        - Continually sample actions, ϵ-soft.
    - Estimation policy: evaluated and improved.
        - Deterministic, greedy.
    - Two policies may be unrelated.

Off-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $N(s,a) \leftarrow 0$; Numerator and
  $D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$
  $\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:
  (a) Select a policy π' and use it to generate an episode:
    $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$
  (b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
  (c) For each pair s, a appearing in the episode at time τ or later:
    $t \leftarrow$ the time of first occurrence of s, a such that $t \geq \tau$
    $w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$
    $N(s,a) \leftarrow N(s,a) + wR_t$
    $D(s,a) \leftarrow D(s,a) + w$
    $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
  (d) For each $s \in \mathcal{S}$:
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$

Off-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
- $Q(s,a) \leftarrow$ arbitrary
- $N(s,a) \leftarrow 0$; Numerator and
- $D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$
- $\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:
- (a) Select a policy π' and use it to generate an episode:
  - $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$
- (b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
- (c) For each pair s, a appearing in the episode at time τ or later:
  - $t \leftarrow$ the time of first occurrence of s, a such that $t \geq \tau$
  - $w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$
  - $N(s,a) \leftarrow N(s,a) + wR_t$
  - $D(s,a) \leftarrow D(s,a) + w$
  - $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
- (d) For each $s \in \mathcal{S}$:
  - $\pi(s) \leftarrow \arg\max_a Q(s,a)$

- Method learns only from tails of episodes.
  - Potentially cause slow learning.

Example: Blackjack

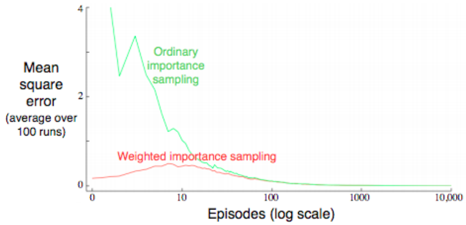
- Estimate value of single state from off-policy data.
  - Dealer is showing 2.
  - Sum of player's cards is 13.
  - Player has usable ace.

Example: Blackjack

- Estimate value of single state from off-policy data.
  - Dealer is showing 2.
  - Sum of player's cards is 13.
  - Player has usable ace.
- Data generated by starting in this state, hit or stick at random with equal probability (behaviour policy).
- Target policy to stick only on sum of 20 or 21.

Example: Blackjack

- Estimate value of single state from off-policy data.
  - Dealer is showing 2.
  - Sum of player's cards is 13.
  - Player has usable ace.
- Data generated by starting in this state, hit or stick at random with equal probability (behaviour policy).
- Target policy to stick only on sum of 20 or 21.



- Optimal value of state under target policy $\approx$ -0.27726.

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment.
  - **No need** for full models.
  - **No need** to learn about ALL states.
    - Less harm by Markovian violations (no bootstrapping).

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment.
  - **No need** for full models.
  - **No need** to learn about ALL states.
    - Less harm by Markovian violations (no bootstrapping).
- MC methods provide alternate policy evaluation process.
  - Average many returns that start in a given state.

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment.
  - **No need** for full models.
  - **No need** to learn about ALL states.
    - Less harm by Markovian violations (no bootstrapping).
- MC methods provide alternate policy evaluation process.
  - Average many returns that start in a given state.
- Control methods and approximating action-value functions.
  - MC intermix policy evaluation and policy improvement.

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment.
  - **No need** for full models.
  - **No need** to learn about ALL states.
    - Less harm by Markovian violations (no bootstrapping).
- MC methods provide alternate policy evaluation process.
  - Average many returns that start in a given state.
- Control methods and approximating action-value functions.
  - MC intermix policy evaluation and policy improvement.
- One issue to watch for: maintaining sufficient exploration.
  - Exploring starts.
  - On-policy and off-policy methods.