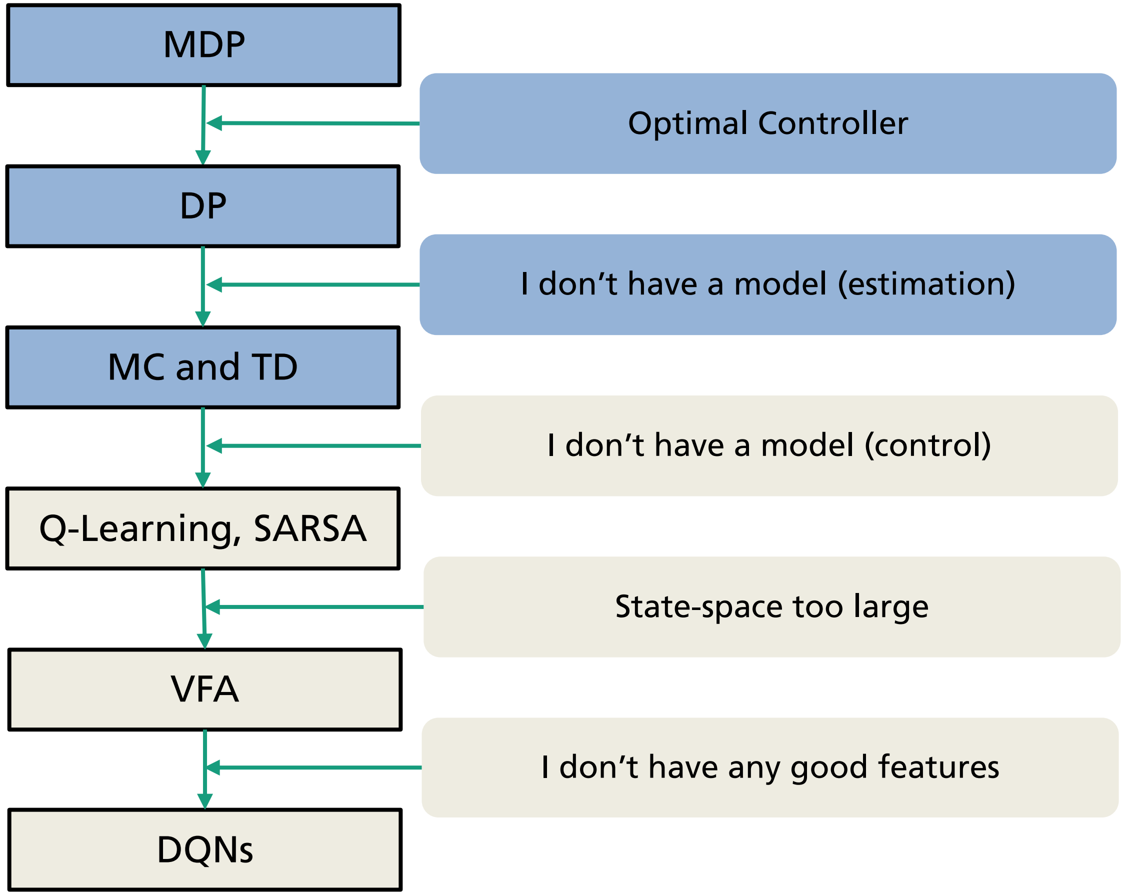
Model-free Prediction: Monte-Carlo

Christopher Mutschler

Overview


MDP
Optimal Controller
DP
I don’t have a model (estimation)
MC and TD
I don’t have a model (control)
Q-Learning, SARSA
State-space too large
VFA
I don’t have any good features
DQNs

Monte Carlo and TD Methods

Assumptions

- We know that the model of the world can be described by an MDP:

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

- We know the (discrete) state and action spaces, i.e., $\mathcal{S}$ and $\mathcal{A}$.

- We can interact with the world (with some policy π).

- We receive experience samples from the environment in the form

$$(S_t, A_t, R_t, S_{t+1}) = (s, a, r, s').$$

Monte Carlo and TD Methods

- Idea:
  - Use the samples to estimate the true V- and Q-value functions for the policy π:

$$V^{\pi}(s)$$
$$Q^{\pi}(s,a)$$

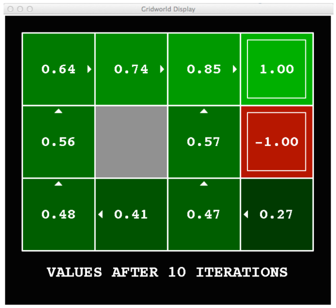
  - Use value function estimations for model-free prediction:

$$V(s) \approx V^{\pi}(s)$$
$$Q(s,a) \approx Q^{\pi}(s,a).$$

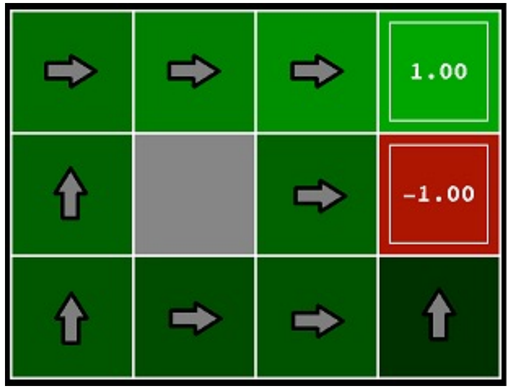
  - Two policy evaluation approaches:
    - Monte Carlo (MC) Learning
    - Temporal Difference (TD) Learning
    - variants in between, i.e., TD(λ)

Monte Carlo and TD Methods

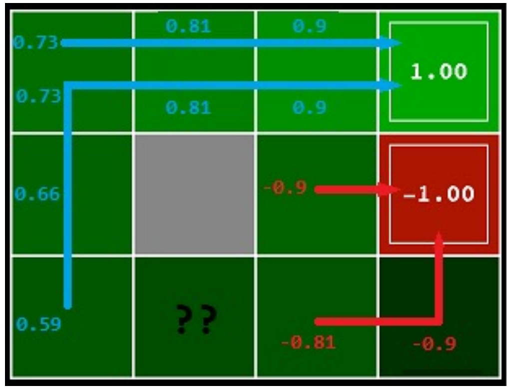
Remember:



- Idea:
  - Use the samples to estimate the true V- and Q-value functions for the policy π



Given Policy



Randomly select state and follow policy & Compute discounted return for each state

Average the values on each state

https://medium.com/@zsalloum/monte-carlo-in-reinforcement-learning-the-easy-way-564c53010511

Monte Carlo Policy Evaluation

- MC Policy Evaluation
  - MC methods learn from episodes of experience under policy π:

$$s_t, a_t, r_t, s_{t+1}, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T \sim \pi$$

  - To evaluate a state $s \in \mathcal{S}$ we keep track of the rewards received from that state onwards.

- First-Visit Monte-Carlo Policy Evaluation:
  - 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: $V(s) = S(s)/N(s)$
  - Our estimation $V(s)$ will come close to $V^{\pi}(s)$ as $N(s) \rightarrow \infty$.
    (considering the law of large numbers)

Monte Carlo Policy Evaluation

- MC Policy Evaluation
  - MC methods learn from episodes of experience under policy π:

$$s_t, a_t, r_t, s_{t+1}, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T \sim \pi$$

  - To evaluate a state $s \in \mathcal{S}$ we keep track of the rewards received from that state onwards.

- Every-Visit Monte-Carlo Policy Evaluation:
  - 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: $V(s) = S(s)/N(s)$
  - Our estimation $V(s)$ will come close to $V^{\pi}(s)$ as $N(s) \rightarrow \infty$.
    (considering the law of large numbers)

Monte Carlo Policy Evaluation

- MC Policy Evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

      Append G to $Returns(S_t)$

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

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo Policy Evaluation

Example: Blackjack. MDP:

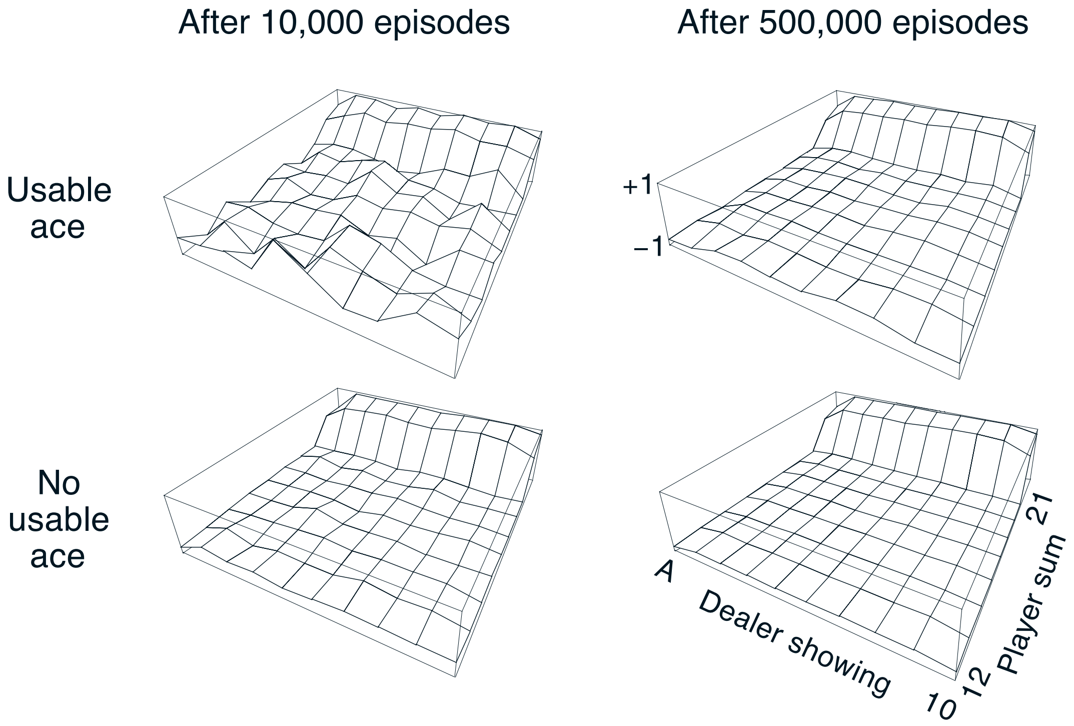
- States:
  - Current sum (12-21) [$\mathcal{P}$ models an automatic twist if sum of cards < 12]
  - Dealer's showing card (ace-10)
  - Do I have a usable ace (yes or no)
- Actions:
  - Stick: stop receiving cards (and terminate)
  - Twist: take another card (no replacement)
- Rewards:
  - Stick:
    - +1 if sum of cards > sum of dealer cards
    - 0 if sum of cards = sum of dealer cards
    - -1 if sum of cards < sum of dealer cards
  - Twist:
    - -1 if sum of cards > 21 (and terminate), 0 otherwise

source: shutterstock.com

Monte Carlo Policy Evaluation

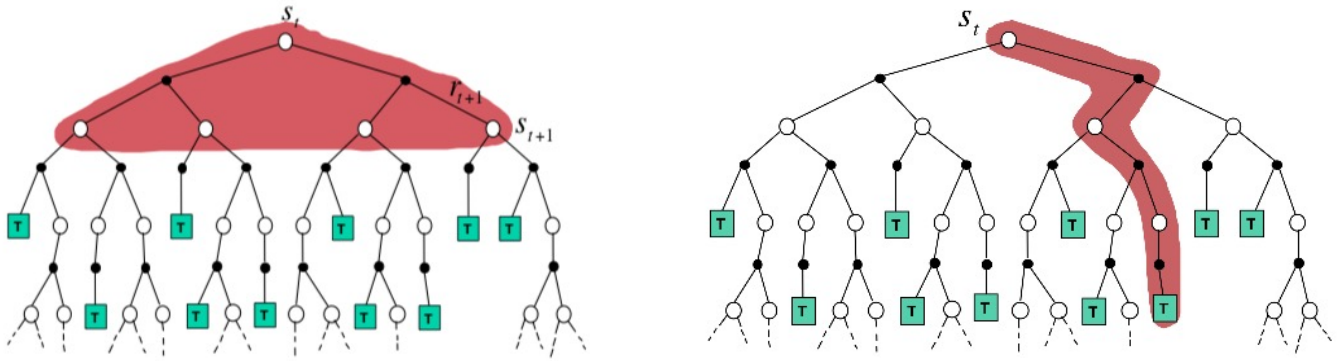
Example: Blackjack

- π: stick if sum of cards ≥ 20 (i.e., 20 or 21), otherwise twist.
- No discounting.
- Cards are dealt with replacement
  (i.e., counting cards does not help)



Monte Carlo Policy Evaluation

- Backup Diagrams compared to DP:



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo Policy Evaluation

- MC Policy Evaluation
- **Incremental Mean**: the mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally:

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\,\mu_{k-1}\right) \\
&= \frac{1}{k}\left(x_k + k u_{k-1} \; - u_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}$$

Monte Carlo Policy Evaluation

- MC Policy Evaluation
- **Incremental Monte-Carlo Updates**

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\,\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

→

- Update $V(s)$ incrementally after each episode.
- For each state s with actual return G:

$$N(s) \leftarrow N(s) + 1$$ (just increment visit counter)

$$V(s) \leftarrow V(s) + \frac{1}{N(s)}\left(G - V(s)\right)$$ (update a bit → reduce error)

- In non-stationary problems, it can be useful to track a running mean, i.e., forget old episodes:

$$V(s) \leftarrow V(s) + \alpha\left(G - V(s)\right).$$

$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$