Theory and Methods for Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 4: Temporal-Difference Learning

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-618 (Spring 2020)

License Information for Reinforcement Learning Slides

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Outline

- This class:
  1. Temporal-Difference Prediction
  2. Temporal-Difference Control
- Next class:
  1. n-step Bootstrapping

Recommended reading

- Chapter 6 in S. Sutton, and G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

Motivation

Motivation

In the previous lecture, we studied the Monte Carlo methods for RL. We have to wait until the termination of an episode to do an MC update. But if it takes too long for completion, what can we do? TD Learning!

Temporal-Difference (TD) Learning

- Combination of Monte Carlo (MC) and Dynamic Programing (DP) ideas:
  1. *model-free* (like MC) – learn directly from experiences, without the knowledge of MDP.
  2. *bootstrap* (like DP) – update estimates based in part on other learned estimates.
  3. *online* – learn from incomplete episodes, without waiting for a final outcome.

MC Prediction

Policy evaluation v_π

For a given policy π, estimate the state value function:

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned}$$

- Constant-α MC update:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

  - G_t is the actual return following time t.
  - α is a constant step-size parameter.
  - must wait until the end of the episode to determine the increment to $V(S_t)$.
  - the target for the MC update is G_t.

DP Prediction

Policy evaluation v_π

For a given policy π, estimate the state value function:

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned}$$

- Iterative DP update:

$$V(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a \mid s)[R(s,a) + \gamma \sum_{s'\in\mathcal{S}} T(s' \mid s,a)V(s')]$$

  - knowledge of the MDP is required.
  - $v_\pi(s')$ is not known and the current estimate, $V(s')$, is used instead.

TD Prediction

Policy evaluation v_π

For a given policy π, estimate the state value function:

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned}$$

- TD(0) update:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

  - update immediately on transition to S_{t+1} and receiving R_{t+1}.
  - need to wait only until the next time step.
  - the target for the TD update is $R_{t+1} + \gamma V(S_{t+1})$.
  - combine the sampling of Monte Carlo with the bootstrapping of DP.

Advantages of TD Prediction Methods

- TD's advantage over DP:
  - model-free – do not require reward or next-state probability distributions.
- TD's advantage over MC:
  - can be implemented in an online, fully incremental fashion.
  - applicable in non-terminating environments as well.

Bias–Variance Tradeoff

- MC target $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-t-1} R_T$ is unbiased.
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is unbiased.
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is biased.
- MC target relies on many random steps, thus have higher variance.
- TD target relies only on the next random step, thus have much lower variance.

TD(0) Prediction Algorithm

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Algorithm Parameter: step size $\alpha \in (0, 1]$

Initialize: $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop for each episode:

Initialize state S

Loop for each step of episode:

$A \leftarrow$ action by policy π for state S

Take action A, observe state S' and reward R

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

until S is terminal

TD Error

Definition (TD Error)

TD error at time t measures the difference between the current estimated value $V(S_t)$ and the better estimate $R_{t+1} + \gamma V(S_{t+1})$:

$$\delta_t \;:=\; R_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

- TD error at each time is the error in the estimate made at that time.
- TD error depends on the next state and next reward, it is not actually available until one time step later.
- that is, δ_t is the error in $V(S_t)$, available at time $t+1$.

MC Error

- The MC error can be written as a sum of TD errors:

$$
\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - V(S_{t+2})) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots + \gamma^{T-t}(G_T - V(S_T)) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots + \gamma^{T-t}(0 - 0) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k
\end{aligned}
$$

- This identity is not exact if $V(S_t)$ is updated during episode.
- It may hold approximately when step size α is small.

Example: Driving Home

State	*Elapsed Time*	*Predicted Time to Go*	*Predicted Total Time*
leaving office	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
secondary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

- the rewards are the elapsed times on each leg of the journey.
- $\gamma = 1$: the return for each state is the actual time to go from that state.
- the value of each state is the expected time to go.
- the second column of numbers gives the current estimated value for each state encountered.

Example: Driving Home

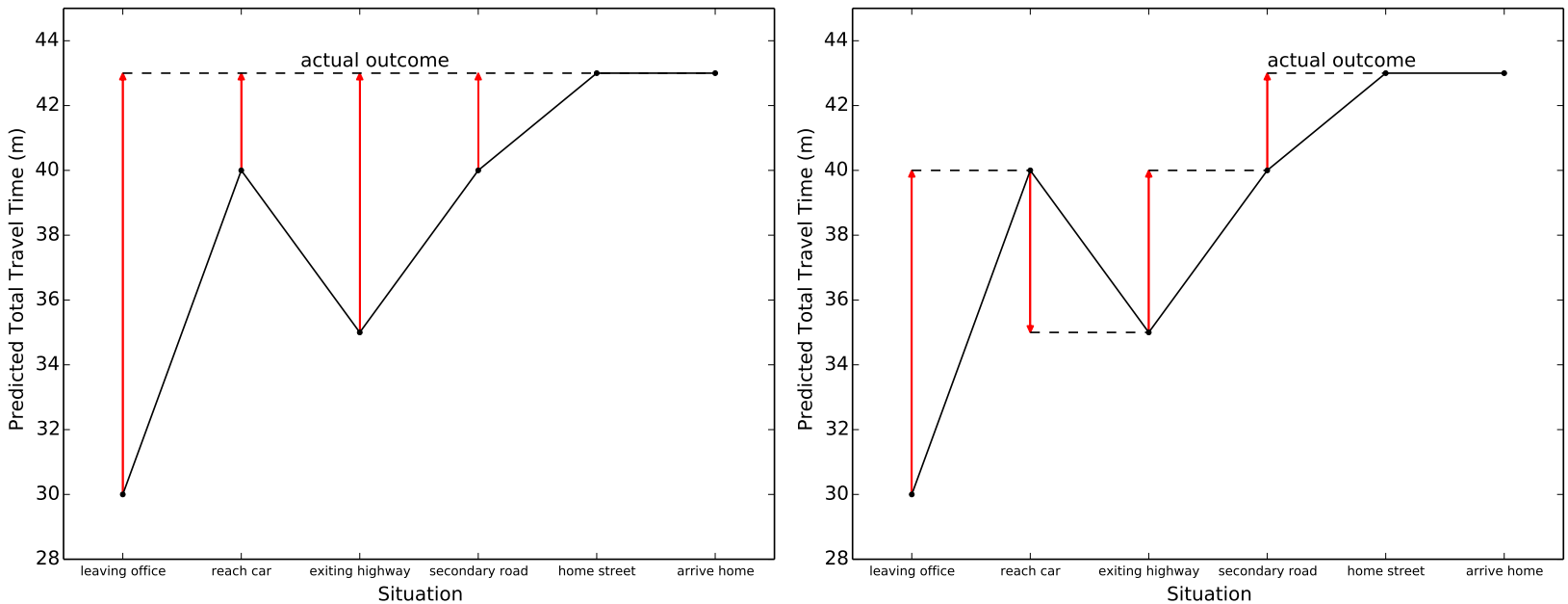


Figure: Changes recommended in the driving home example by MC methods (left) and TD methods (right) with $\alpha = 1$.

Convergence of TD(0)

Theorem (Convergence of TD(0))

For any fixed policy π, TD(0) methods will converge to values of v_π, in the mean for a constant step-size α if it is sufficiently small, and with probability 1 if the decreasing step-size $\{\alpha_t\}$ satifies the following condition:

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

- Which one of MC and TD methods converge faster?

An open problem, but we can compare them empirically.

Example: Random Walk

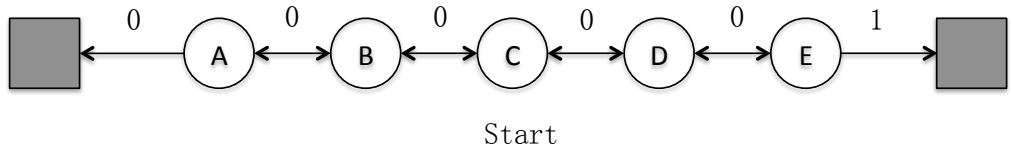


Figure: Random walk and the reward of each transit.

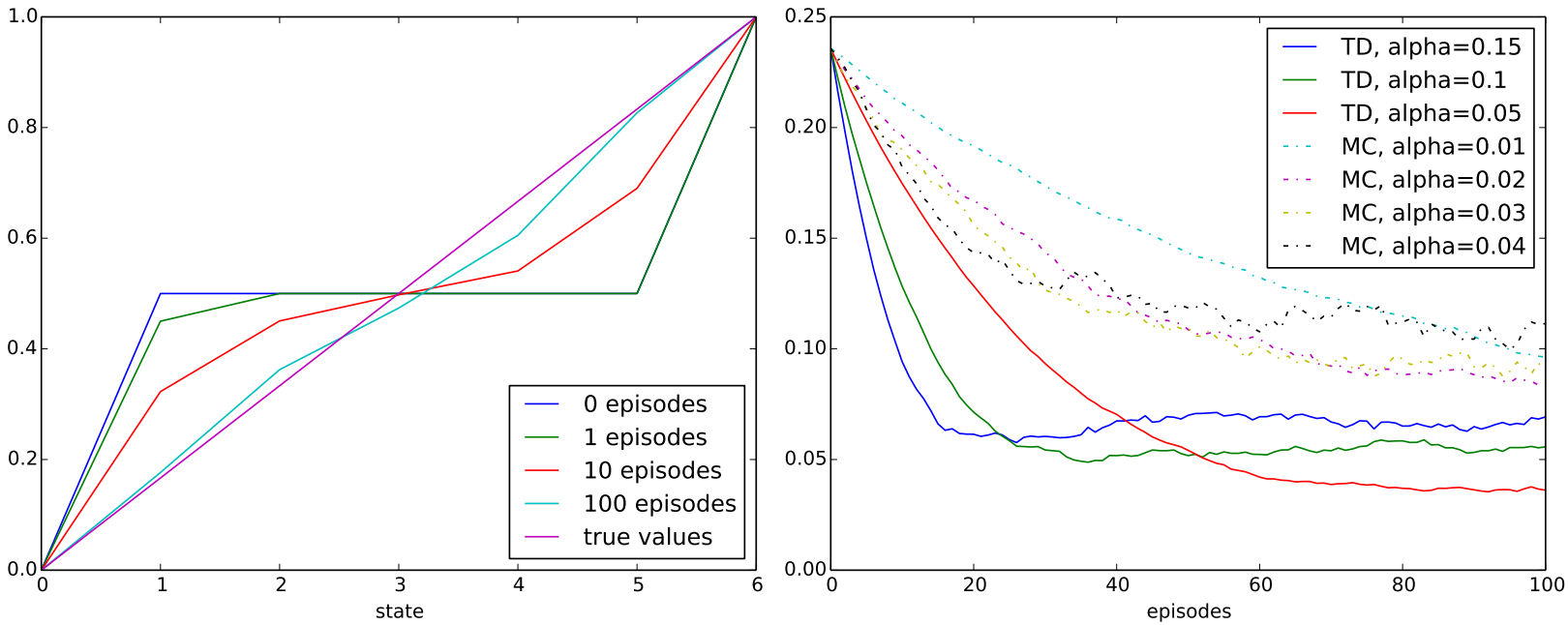


Figure: Left: values learned after various number of updates in a single run of TD(0). Right: the root mean-squared (RMS) error between the value functions learned and the true values.

Batch Update of MC and TD

• In some cases, there are only limited number of experiences available.

• Batch update calculates the increments in each times step but updates the value function only after processing each complete batch of training data.

• Under batch updating, both TD and MC methods with sufficiently small step size will converge deterministically, but not necessarily to the same point.

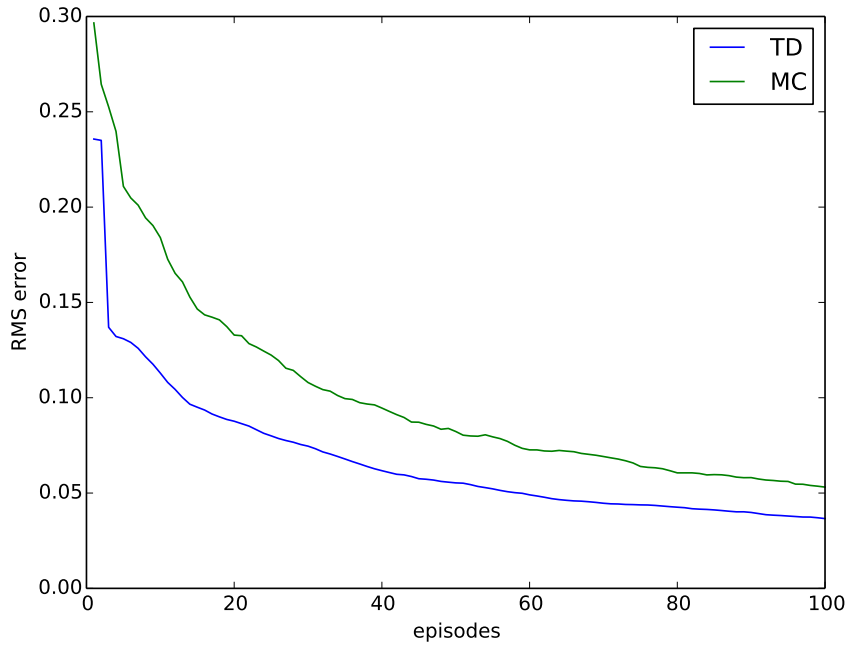


Figure: The RMS error of MC and TD batch updating in the random walk example.

Example: You are the Predictor

- States $\mathcal{S} = \{A, B\}$.
- No discounting: $\gamma = 1$.
- 8 episodes of experience:

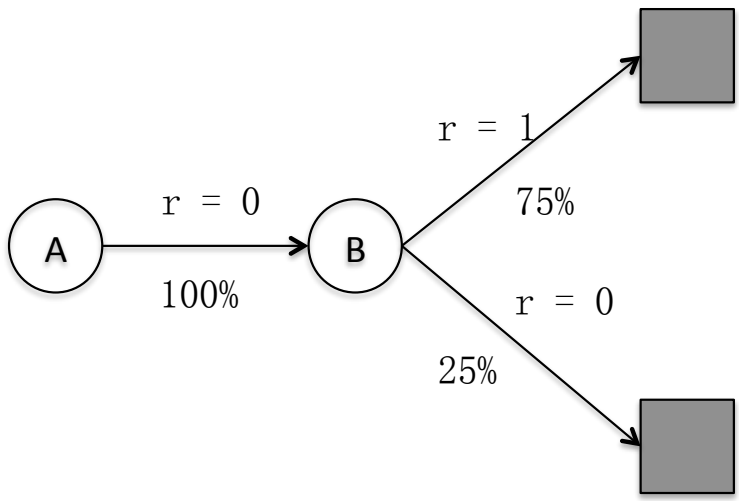
$$A, 0, B, 0 \quad B, 1 \quad B, 1 \quad B, 1$$
$$B, 1 \quad B, 1 \quad B, 1 \quad B, 0$$

- What is the value of $V(A)$?

Example: You are the Predictor

- MC methods: we have seen A only once and the reward is 0, so $V(A) = 0$.

- TD methods: 100% of the time A traverses to B, whose estimated value is 0.75, so $V(A) = 0.75$.



Optimality of TD(0)

Definition (Certainty-Equivalence Estimate)

Given some observed episodes, a Markov process can be constructed in the following way:

- the transition probability from state A to B is the fraction of observed transition from A that goes to B.
- the associated reward is the average reward obtained on those transitions.

Given this model, *certainty-equivalence estimate* is the value function which can be computed exactly.

- In general, TD(0) converges to the certainty-equivalence estimate.

Optimality of TD(0)

- TD method exploits Markov property:
  - batch TD update converges to solution of maximum likelihood Markov model.
  - more effective in Markov environments.
- MC method does not exploit Markov property:
  - batch MC update converges to solution with minimum mean-squared error on observed experience.
  - more effective in non-Markov environments.

TD Control

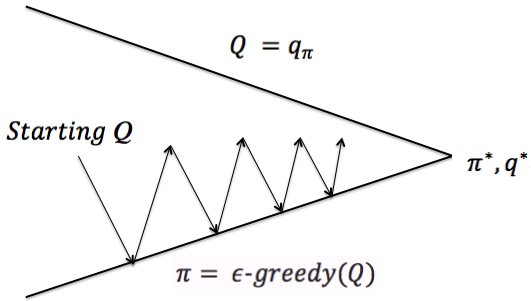
- On-policy methods: SARSA.
- Off-policy methods: Q-learning.

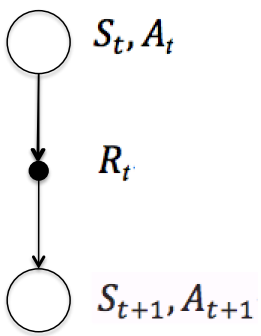
SARSA: On-policy TD Control

• On-policy methods alternately estimate q_π for the behavior policy π, and update π towards the greediness with respect to q_π.

• SARSA considers transitions from one state-action pair to another state-action pair:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$





SARSA Algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$

Initialize: $Q(s, a)$, for all $s \in \mathcal{S}^+$ and $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize state S

Choose A from S using policy based on Q (e.g., ϵ-greedy)

Loop for each step of episode:

Take action A, observe reward R and next state S'

Choose A' from S' using the policy based on Q (e.g., ϵ-greedy)

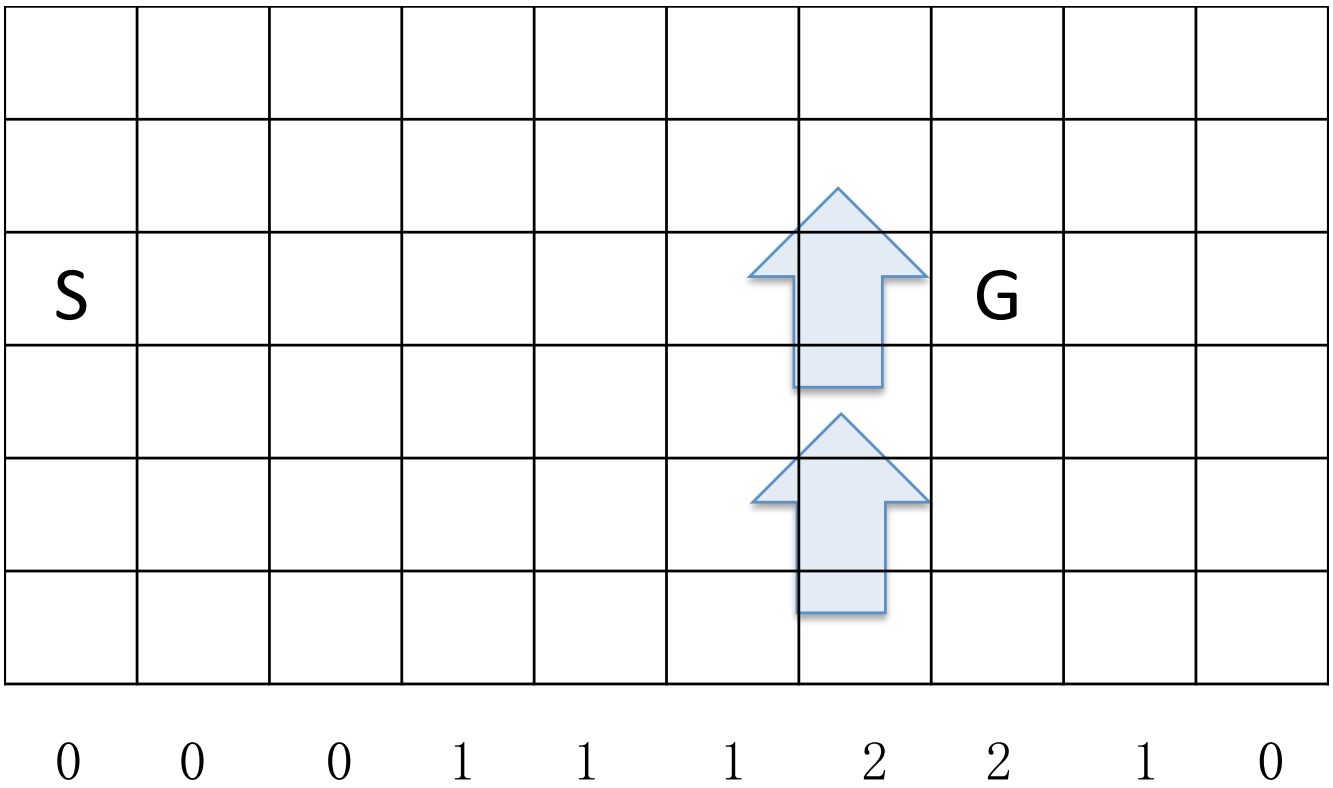
$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'$; $A \leftarrow A'$

until S is terminal

Example: Windy Gridworld

- Actions $\mathcal{A} = \{\text{up}, \text{down}, \text{right}, \text{left}\}$
- This is an undiscounted episodic task.
- Constant rewards of -1 until the goal state is reached.



Example: Windy Gridworld

• The increasing slope of the graph shows that the goal was reached more quickly over time.

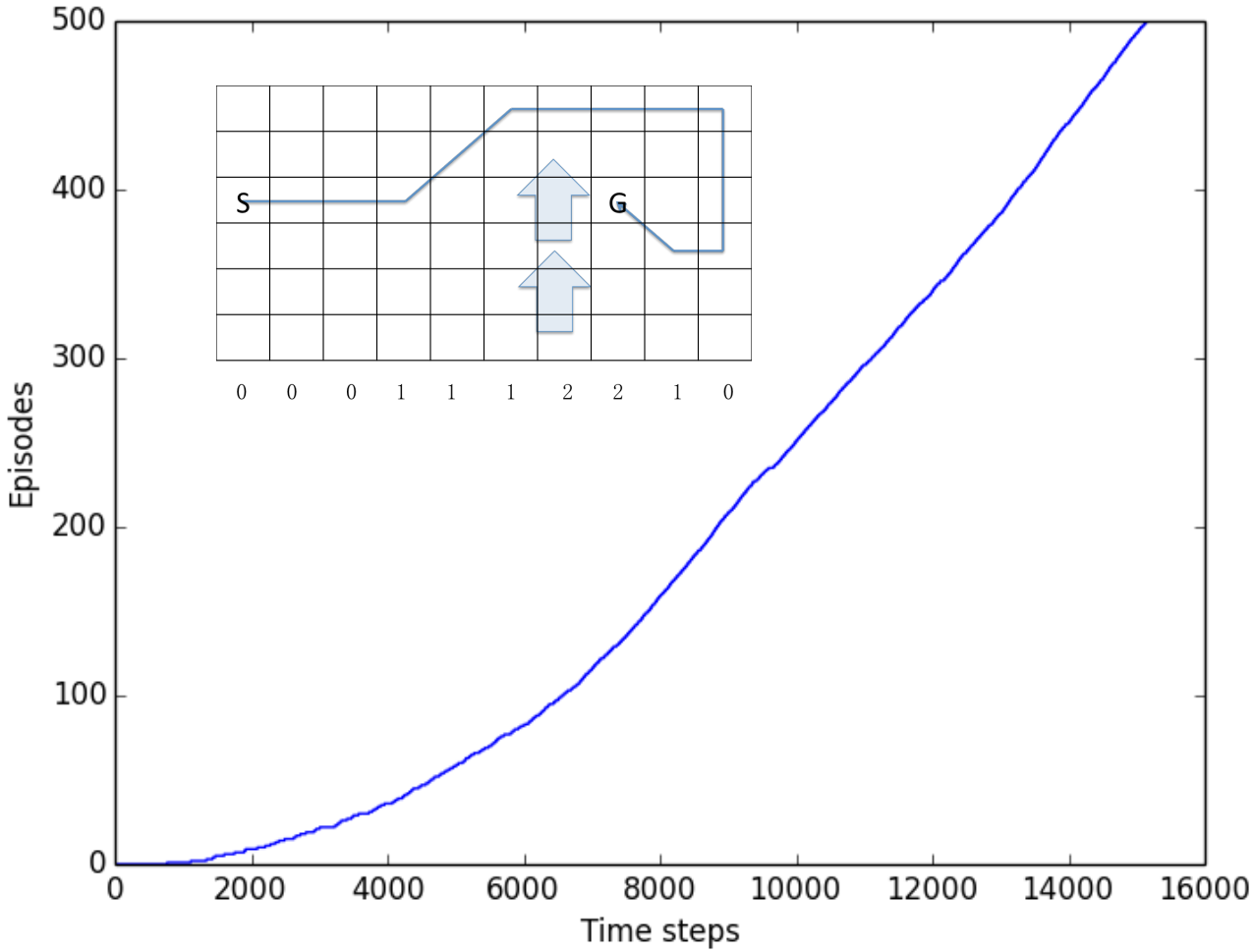


Figure: Time step - episode curve with SARSA $\epsilon = 0.1$, $\alpha = 0.5$

Q-learning: Off-policy TD Control

- Q-learning update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

- Off-policy methods evaluate the policy π while using another behavior policy μ.

- The learned action-value function, Q, directly approximates q_*, the optimal action-value function.

- The policy still has an effect in that it determines which state-action pairs are visited and updated.

- Q has been shown to converge with probability 1 to q_*, under the assumptions:
  - all pairs continue to be updated, and
  - a variant of the usual stochastic approximation conditions on the sequence of step-size parameters.

Q-learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$

Initialize: $Q(s, a)$, for all $s \in \mathcal{S}^+$ and $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize state S

Loop for each step of episode:

Choose A from S using policy based on Q (e.g., ϵ-greedy)

Take action A, observe reward R and next state S'

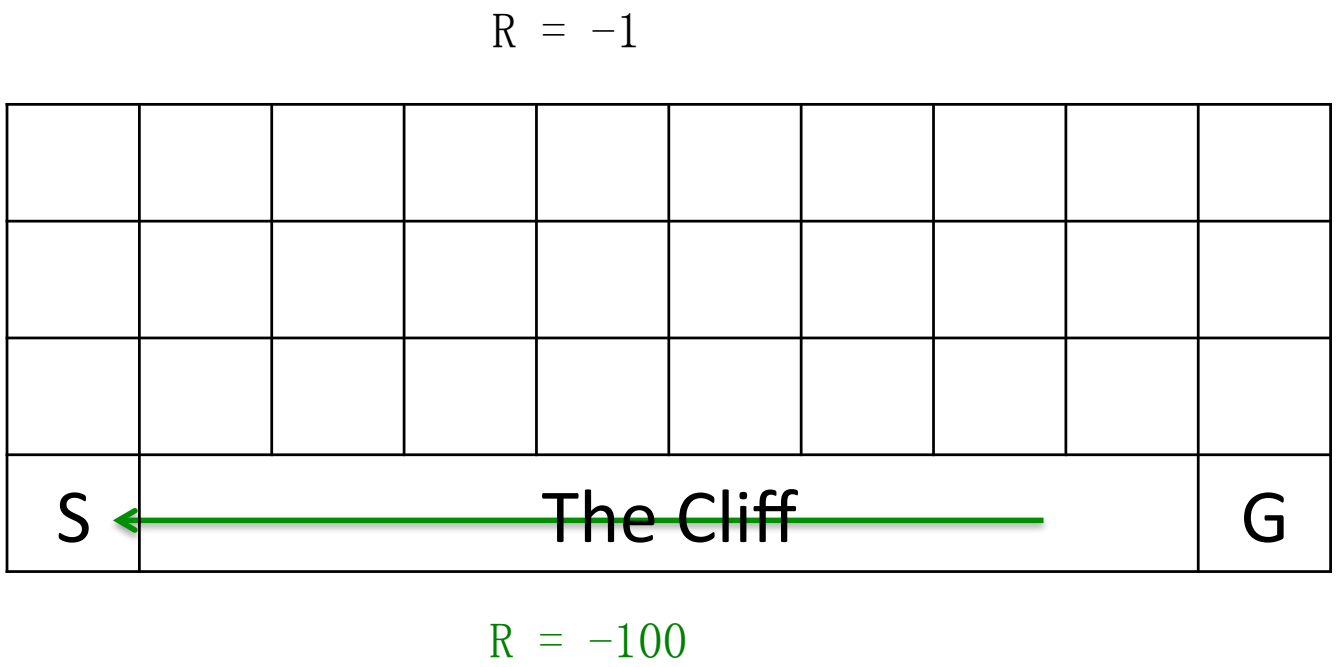
$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

until S is terminal

Example: Cliff Walking

- Actions $\mathcal{A} = \{\text{up}, \text{down}, \text{right}, \text{left}\}$.
- This is an undiscounted episodic task.
- The reward is -1 on all transitions except the cliff region.
- Stepping into the cliff region incurs a reward of -100 and sends the agent instantly back to the start.



Example: Cliff Walking

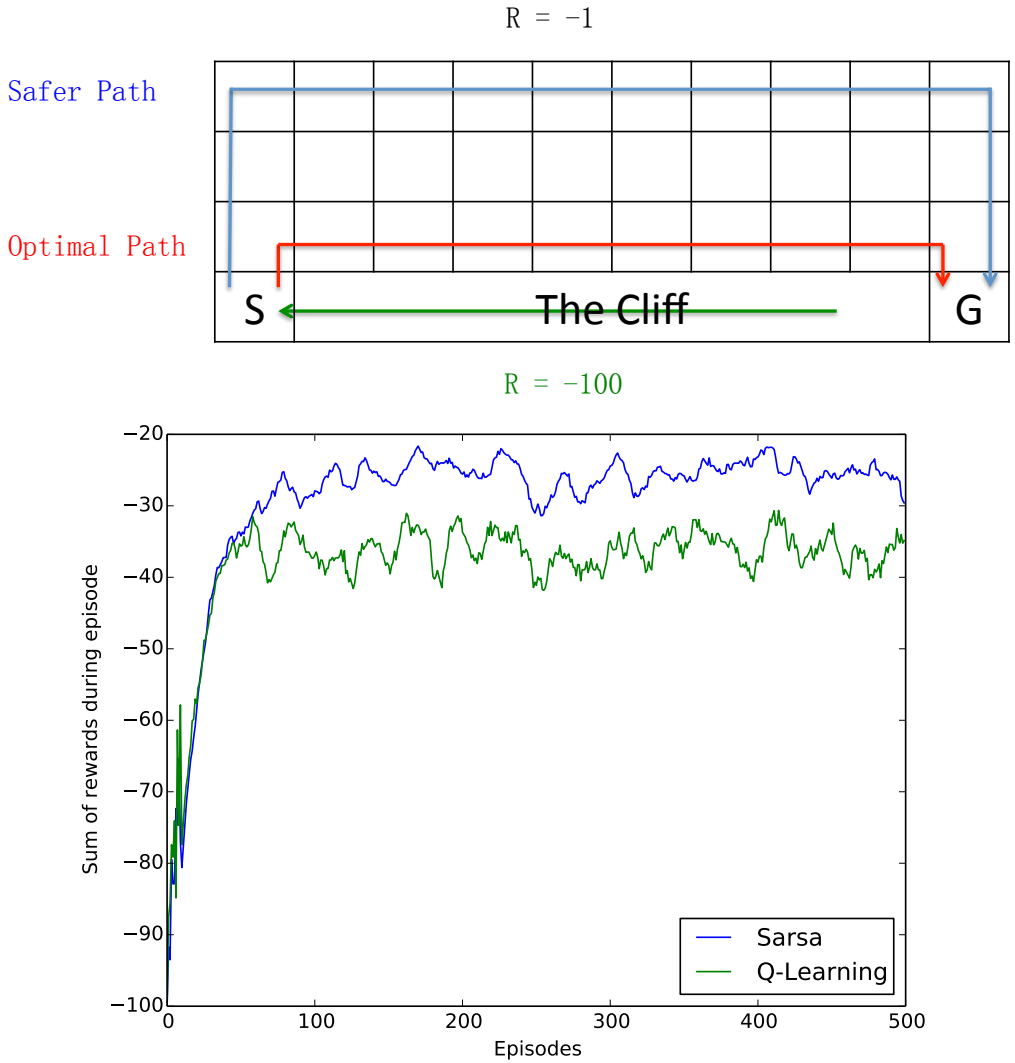


Figure: (Top) Safer path and optimal path.
(Bottom) Online performance of SARSA and Q-learning.

Expected SARSA

• Update similar to Q-learning except that it uses the expectation instead of the maximum over the actions:

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\{R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\} \\ &\leftarrow Q(S_t, A_t) + \alpha\{R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\} \end{aligned}$$

• This algorithm moves deterministically in the same direction as SARSA moves in expectation.

• Expected SARSA has more complex computation but, in return, it has smaller variance and generally leads to better performance compared with SARSA.

Example: Cliff Walking

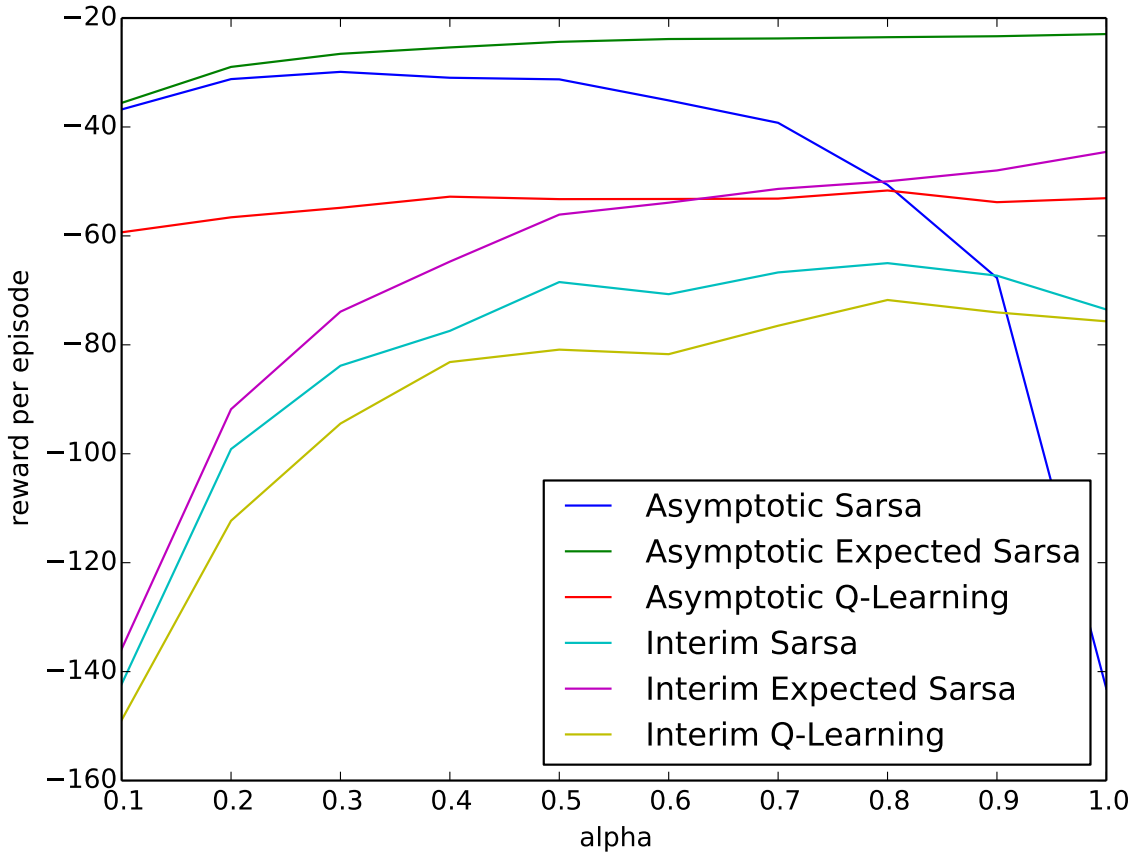
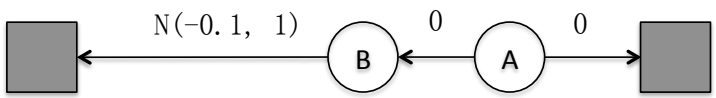


Figure: Asymptotic and interim performance of SARSA, Expected SARSA and Q-learning, with various values of α and $\epsilon = 0.1$. Asymptotic performance is the average of 100000 episodes while interim performance is the average of first 100 episodes. These data are averages of over 50000 and 10 runs for the interim and asymptotic cases respectively.

Maximization Bias

- SARSA and Q-learning both involve maximum in the construction of their target policies. (greedy and ϵ-greedy).

- Maximization operator leads to positive bias!



Double Learning

• Maximization bias arises because we use the same sample to determine the maximization action and estimate its value.

• Decouple the choice of maximization action and its estimation – learn two independent estimates: $Q_1(a)$ and $Q_2(a)$

• Given $A^* = \arg\max_a Q_1(a)$, we have:

$$\begin{aligned} \mathbb{E}[Q_1(A^*)] &\geq q(A^*) \\ \mathbb{E}[Q_2(A^*)] &= q(A^*) \end{aligned}$$

Double Q-Learning Algorithm

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$

Initialize: $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+$ and $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize state S

Loop for each step of episode:

Choose A from S using policy based on $Q_1 + Q_2$ (e.g., ϵ-greedy)

Take action A, observe reward R and next state S'

With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha[R + \gamma Q_2(S', \arg\max_a Q_1(S', a)) - Q_1(S, A)]$$

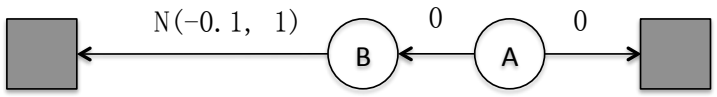
Else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha[R + \gamma Q_1(S', \arg\max_a Q_2(S', a)) - Q_2(S, A)]$$

$S \leftarrow S'$

until S is terminal

Q-Learning v.s. Double Q-Learning



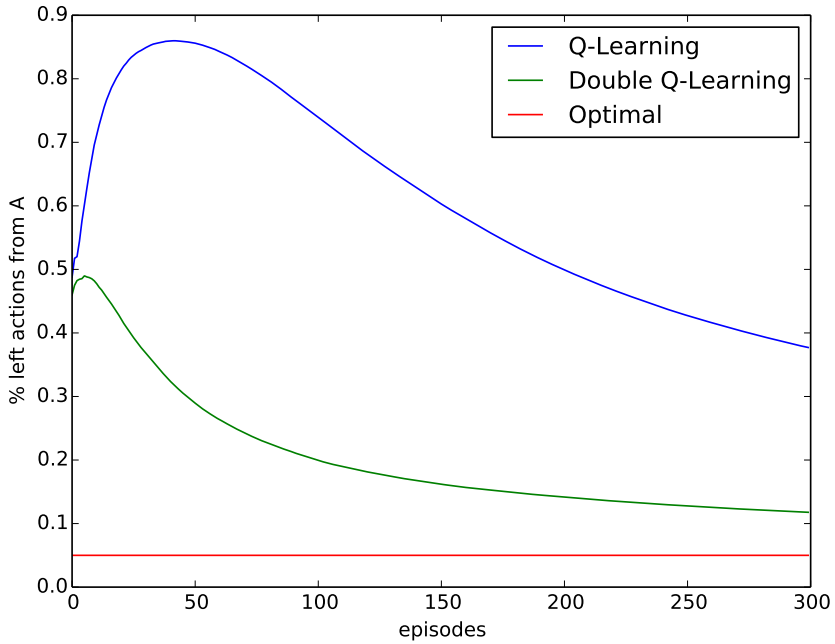


Figure: Comparison between Q-learning and double Q-learning. Q-learning initially learns to take the left action much more often than the right action.

Summary - DP, MC and TD

Methods	**DP**	**MC**	**TD**
Model knowledge	Need	No need	No need
When do updates	After next step	After whole episode	After next step
Bias	-	Unbiased	Biased
Variance	-	Big	Small
Convergence in batch update	-	min mean-squared error	max likelihood Markov model

Summary - SARSA, Q-learning

- SARSA: on-policy control, behavior policy is the same as the policy to estimate.
- Q-learning: off-policy control, behavior policy is different as the policy to estimate.
- Expected SARSA: More computation, less variance, empirically better performance.
- Double Learning: decouple action choice and estimation to avoid maximization bias.