Temporal Difference Learning

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Advanced Machine Learning
September–December 2021

Adding bootstrapping to Monte Carlo methods

- Dynamic programming: use generalized policy iteration to approximate π_*, v_*
  - Bootstrap from an initial estimate through incremental updates
  - Need to know the model
- Monte Carlo methods: random exploration to estimate π_*, v_*
  - Works with black box models
  - Need to complete an episode before applying updates
- Temporal Difference (TD) learning
  - Apply bootstrapping to Monte Carlo methods

From Monte Carlo to TD

- Monte Carlo update for non-stationary environments
  - $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$, $\alpha \in (0,1]$ is a constant
  - G_t is available only after we complete the episode — calculate backwards from G_T
- Instead
  - Expand G_t as $R_{t+1} + \gamma V(S_{t+1})$
  - Revised update rule: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
  - R_{t+1} is available after choosing A_t
  - Use current estimate for $V(S_{t+1})$
  - Update $V(S_t)$ on the fly, as the episode evolves
- Also called TD(0), because it has zero lookahead
  - More generally, can look ahead n steps to update, TD(n)
  - Most general version is called TD(λ), we only consider TD(0)

TD(0) algorithm for policy evaluation

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 $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

TD(0) example: Driving home from work

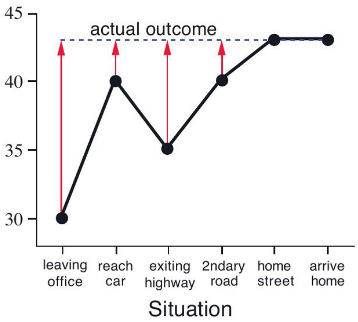
- Predict how long it will take you to drive home from work
- Leave office on Friday at 6:00 pm, initial estimate 30 minutes from now
- Reach car at 6:05 pm, raining, revise estimate to 35 minutes from now, total 40
- At 6:20 pm, complete highway stretch smoothly, cut estimate of total to 35 minutes
- Stuck behind slow truck, follow till 6:40 pm
- Turn off onto home street, arrive at 6:43 pm

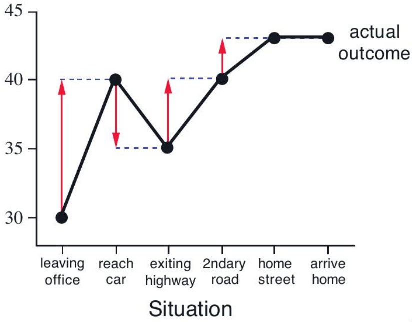
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

TD(0) example: Driving home

- Rewards: elapsed time on each leg
- No discounting: $\gamma = 1$, return at a state is actual time remaining

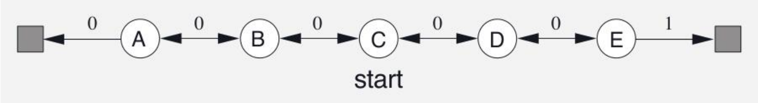
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43



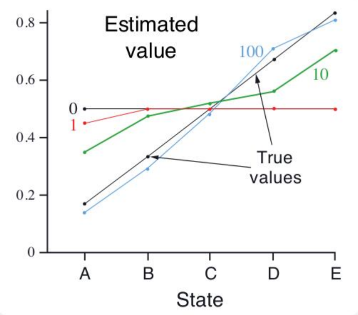


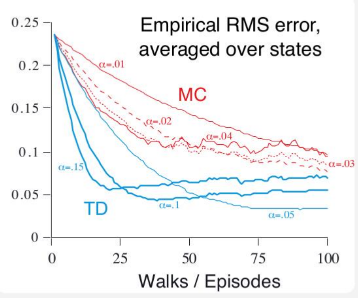
Comparing MC and TD(0)

- Markov Reward Process



- Reward is probability of reaching right hand side



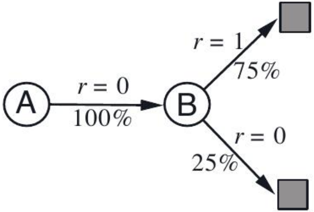


Comparing MC and TD(0) ...

Predict the values of states A and B

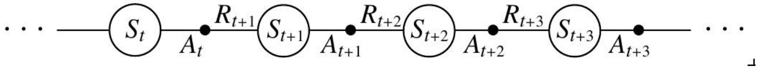
A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

- $V(B) = 6/8 = 0.75$
- What about $V(A)$?
- MC — one episode, $V(A) = 0$
- TD(0) – $V(A) = 0.75$



SARSA: On policy TD control, estimating π_*

- For π_*, better to estimate q_π rather than v_π
- Structure of an episode



- Use the following update rule

  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
- Update uses $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, hence the name SARSA
- As with Monte Carlo estimation, use ε-soft policies to balance exploration and exploitation

SARSA algorithm on-policy TD control

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

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning: Off policy TD control, estimating π_*

- Directly estimate q_* independent of policy being followed
- Use the following update rule
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
- Underlying policy still needs to be designed to visit all state-action pairs
- With suitable assumptions, Q-learning provably converges to q_*

Q-learning algorithm, off-policy TD control

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

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Summary

- Temporal difference methods combine bootstrapping with Monte Carlo exploration of state space
- SARSA is a TD(0) algorithm for on-policy control — estimating π_*
- Q-learning is an off-policy algorithm that provably converges to q_*
- TD-based approaches apply beyond reinforcement learning
  - General methods to make long term predictions about dynamical systems
- Theoretical properties such as convergence still an area of research