Temporal-Difference Learning
Week #6

Introduction

- *Temporal-Difference (TD) Learning*
  - a combination of DP and MC methods
    - updates estimates based on other learned estimates (i.e., *bootstraps*), (as DP methods)
    - does not require a model; learns from raw experience (as MC methods
  - constitutes a basis for the reinforcement learning.
  - Convergence to V^{π} is guaranteed (asymptotically as in MC methods)
    - in the mean for a constant learning rate α if it is sufficiently small
    - with probability 1 if α decreases in accordance with the usual stochastic approximation conditions.

Update Rules for DP, MC and TD

- DP update

$$V(s_t) = \sum_{a} \pi(s_t, a) \sum_{\substack{adjacent \\ states}} P^a{}_{s_t s_{t+1}} \left[R^a{}_{s_t s_{t+1}} + \gamma V_t(s_{t+1}) \right]$$

current value estimate of next state

- MC update

$$V(s_t) = V(s_t) + \alpha \left[R_t - V(s_t) \right]$$

target

- TD update

$$V(s_t) = V(s_t) + \alpha \left[r_{t+1} + \gamma V_t(s_{t+1}) - V(s_t) \right]$$

most recent return

Update Rules for DP, MC and TD

- *General update rule*:
  - *NewEst = OldEst + LearningParameter (Target - OldEst)*
- *DP update*
  - The new estimate is recalculated at every time step using the information of the completely defined model.
- *MC update*
  - The target in MC methods is the *real average return* obtained at the end of an episode.
- *TD update*
  - The target is the *most recent return* of the environment added to the current estimated value of the next state.

Algorithm for *TD(0)*

- *Initialize V(s) arbitrarily, π to the policy to be evaluated*
- *Repeat for each episode*
  - *Initialize s*
  - *Repeat for each step of episode*
    - *a ← action generated by π for s*
    - *Take action a, observe reward r, and next state s'*

$$V(s) = V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$$

    - *s ← s'*
  - *Until s is terminal*

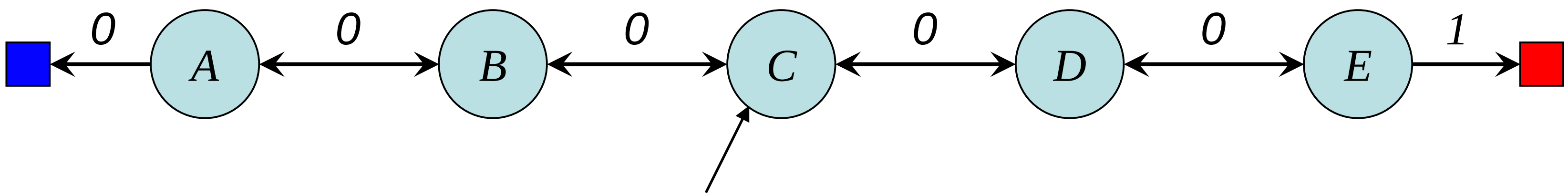
Advantages of TD Prediction Methods

- They *bootstrap* (i.e., learn a guess from other guesses)
  - Question: Learning a guess from a guess still guarantees a convergence to optimal state values?
  - Answer: Yes,
    - in the mean for a constant and sufficiently small α, and
    - certainly (i.e., wp1) if α decreases according to the usual stochastic approximation conditions.
- They need *no model of environment*
  - an advantage over DP methods
- They do *not wait until end of episode* to update state/action values

Advantages of TD Prediction Methods... (2)

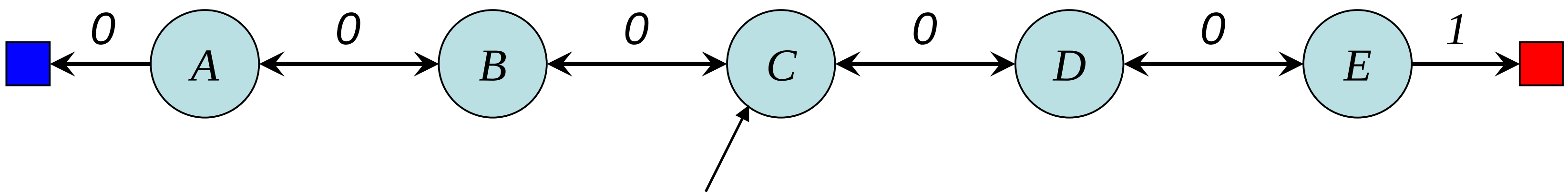
- Constant-α MC and TD methods *bootstrap* and guarantee convergence.
- *Question*: Which methods converge first?
  - There is no mathematical proof answering that question
  - In practice, however, *TD methods are shown to converge usually faster than constant-α MC methods on stochastic tasks*.

Example: Random Walk



- Episodic, no discounted RL task
- States: *A*, ..., *E*;
- Available actions: *left (L)*, *right (R)*
- Goal: *red square*
- Termination: *Both squares*
- Values: ?

Example: Random Walk



- True values:
  - V(A)=1/6; V(B)=2/6; V(C)=3/6; V(D)=4/6; V(E)=5/6;

Batch Updating

- *Motive*: In case the amount of experience is limited, an alternative solution in incremental learning is to repeatedly present experience until convergence.
- *Batch updating* is the name since the updates are performed but recording is postponed until after the entire data are processed.
- Comparing constant-α MC and TD methods conducting random walk experiment under batch updating, we observe TD methods converge faster.

SARSA: On-Policy TD Control

- GPI using TD methods: two approaches
  - On-policy
  - Off-policy
- Transitions are between state-action pairs.
- Action value functions
- Value updates:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

Algorithm: SARSA

- *Initialize Q(s,a) arbitrarily*
- *Repeat for each episode*
  - *Initialize s;*
  - *Choose a from s using policy derived from Q*
  - *Repeat for each step of episode*
    - *Take action a, observe reward r, and next state s'*
    - *Choose a' from s' using policy derived from Q(e.g., ε-greedy)*

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

    - *s* $\leftarrow$ *s'; a* $\leftarrow$ *a';*
  - *Until s is terminal*

Q-Learning: Off-Policy TD Control

- Learned action-value function, Q, directly approximates optimal action-value function, Q^*, independent of the *behavior policy*.
- This simplifies the analysis of the algorithm.
- For convergence, all that the behavior policy is required to do is that it sees the state-action pairs are continuously visited and updated .

Algorithm for Q-learning

- *Initialize Q(s,a) arbitrarily*
- *Repeat for each episode*
  - *Initialize s;*
  - *Repeat for each step of episode*
    - *Choose a from s using policy derived from Q(e.g., ε-greedy)*
    - *Take action a, observe reward r, and next state s'*

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

    - $s \leftarrow s'$;
  - *Until s is terminal*
- *Until convergence occurs*

References

- [1] Sutton, R. S. and Barto A. G., "*Reinforcement Learning: An introduction*," MIT Press, 1998