Reinforcement Learning

Monte Carlo and Temporal Difference Learning
Monte Carlo Methods

- Dynamic Programming
  - Requires complete knowledge of the MDP
  - Spends equal time on each part of the state space
    - In sparse state spaces many states are irrelevant
  - Complexity increases with the number of states (n) and the length of episodes (k)
    - Policy-specific value function: $O(n^3)$
    - Optimal policy value function: $O(n^2 \times k)$

- If model parameters are not known we can use Monte Carlo methods using samples
Monte Carlo Methods

- Monte Carlo methods use random samples
  - Sample trajectories are generated according to the transition probabilities (and a fixed policy)
  - Averaging accumulated value of trajectories originating from a given state provides an approximation of the value of the state

\[
V^\pi(s) \approx \sum_{(s_0a_0r_0,\ldots,s_ka_kr_k)\mid s_0=s} \frac{1}{N_s} \sum_{t=0}^{k} \gamma^k r_k
\]

\[
N_s = \left| \left\{ (s_0a_0r_0,\ldots,s_k a_k r_k) \mid s_0 = s \right\} \right|
\]
Temporal Difference Methods

- Simple Monte Carlo methods use random samples of entire trajectories
  - Value function learned is the one for the policy used to generate the samples
  - Learning of values only after the entire trajectories are generated

- Temporal Difference methods use an estimate of the state value to bootstrap
  - Learning from single transitions
  - More efficient use of the Markov assumption
Temporal Difference Methods

- Temporal Difference methods use random sampling of transitions to update value estimate based on the previous estimate.

- At each step one state value estimate is updated using the TD error:
  \[
  V^\pi(s_t) \leftarrow (1 - \alpha)V^\pi(s_t) + \alpha \left( r_t + \gamma V^\pi(s_{t+1}) \right)
  \]

- Fully incremental
Simple Monte Carlo vs. Temporal Difference Methods

- TD methods are fully incremental
  - Learn before the entire outcome is known
  - Learn from incomplete sequences

- TD and MC converge given certain assumptions on $\alpha$
  - If samples fully represent the Markov Chain they will converge to the same solution
    - Generally, TD will converge faster
  - If samples are biased they will converge to different solutions
    - MC converges to best estimate over samples independent of state (and thus Markov assumption)
    - TD will converge to value of the best fitting Markov Model
Solving MDPs

- Simple MC and TD can learn the value function for the policy used for sampling
  - To learn optimal policy it is necessary to estimate value of the optimal policy.
    - Need to determine how to get improved policy value
      \[
      V'(s) = \max_a \left( R(s) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right)
      \]
    - Either need to have a separate way to estimate policy improvement
    - Or need to remove the max from the value improvement by limiting action choices to one
Actor-Critic Approach

- Actor-Critic systems use a separate learner to estimate the optimal policy
  - Actor: executes actions according to a policy estimate and an exploration strategy
    - Learns to estimate the optimal policy using feedback from the critic
  - Critic: learns the value function of the policy executed by the actor
    - Provides feedback to the actor in the form of the TD-error
Actor-Critic Approach

- Critic uses TD-learning to estimate the state value function of the actor’s policy
  - Critic feedback is the difference between the expected value of the outcome of the policy and the outcome of the action taken by the actor
    \[
    \varepsilon(s,a) = r + \gamma V^\pi(s') - V^\pi(s)
    \]
- Actor uses the feedback to update its policy
  \[
  \pi(s,b) = \begin{cases} 
    \xi \max(0, \pi(s,b) + \beta \varepsilon(s,a)) & b = a \\
    \xi \pi(s,b) & b \neq a 
  \end{cases}
  \]
  \[
  \xi = \max(0, \pi(s,b) + \beta \varepsilon(s,a)) + \sum_{b \neq a} \pi(s,b)
  \]
Actor-Critic Approach

- Actor-Critic systems will only converge under certain conditions
  - Critic has to have a correct estimate of the value of the actor’s current policy
    - Actor has to largely execute the policy that it has learned (on-policy)
    - Critic has to have enough time to adapt its estimate to the changes in the (non-stationary) policy of the actor
  - Critic has to learn significantly faster than the actor
Direct Optimal Value Function Estimation

- Actor-Critic methods approximate the optimal evaluation function using policy improvement.
- Estimating the optimal state value function directly only works if we know the optimal policy.
  - If there is only one possible choice in each state then we can directly estimate the optimal value function.
    - We can treat the action as part of the state.
State/Action Value Functions

- State/Action Value functions, $Q^\pi(s, a)$, represent the value of the outcome of taking action $a$ in state $s$ and then following policy $\pi$:

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P(s' | s, a) V^\pi(s')$$

- State value depends on policy in the state:

$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a)$$

  - For deterministic policies

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

- Temporal difference sampling leads to

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left( R(s) + \gamma \sum_b \pi(s', b) Q^\pi(s', b) - Q^\pi(s, a) \right)$$
State/Action Value Functions

- State/Action value function for the optimal policy
  \[ Q^*(s, a) = R(s) + \gamma \sum_{s'} P(s' | s, a) V^*(s') \]
  - Since there is a deterministic optimal policy the state value is the value of the best action choice
    \[ V^*(s) = \max_a Q^*(s, a) \]
  - The optimal state/action value function is
    \[ Q^*(s, a) = R(s) + \gamma \sum_{s'} P(s' | s, a) \max_b Q^*(s', b) \]
State/Action Value Functions

- \textit{max} is no longer part of the sampling average but of the sample value estimate
  - Can use Temporal Difference sampling to estimate
    \[
    Q(s, a) \leftarrow Q(s, a) + \alpha \left( R(s) + \gamma \max_b Q(s', b) - Q(s, a) \right)
    \]
  - If \(Q(s, a)\) converges (no longer changes) it is the optimal value function \(Q^*(s, a)\)
  - Optimal policy can be directly extracted
    \[
    \pi^*(s) = \arg\max_a Q^*(s, a)
    \]