Reinforcement Learning David Silver - Lecture 4 Notes: Model-Free Prediction

Name: Eli Andrew

- Model-free prediction is the same case as before except that now no one is giving us the MDP.
- Model-free prediction methods go directly from environment interactions (observations) to value-functions without the use of a model (MDP).
- In other words, we are trying to estimate the value function of an *unknown* MDP.
- **Monte-Carlo Learning Overview:**
  - MC methods learn directly from episodes of experience
  - MC is *model-free*: no knowledge of MDP transitions or rewards
  - MC learns on complete episodes: no bootstrapping
  - MC uses simplest possible idea: value = mean return
  - Caveat: can only apply MC to episodic MDPs (all episodes must terminate)
- **Monte-Carlo Learning Details:**
  - Goal: learn v_π from episodes of experience under policy π $(S_1, A_1, R_2, \dots, S_k\ \pi)$
  - Recall that return is the total discounted reward: $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$
  - Recall that the value function is the expected return: $v_\pi(s) = E\pi[G_t | S_t = s]$
  - MC policy evaluation uses *empirical mean return* instead of expected return.
- **First-visit Monte-Carlo Policy Evaluation:**
  - To evaluation state s
  - The *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 $V(s) = \frac{S(s)}{N(s)}$
  - By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$
  - *Note:* $N(s)$ and $S(s)$ persist over all episodes
  - **Important:** what we care about here is ensuring that we visit all states in S that we care about for policy π and not necessarily seeing all states. The way to ensure that we see all states that we care about for policy π is to actually just follow policy π and work with the samples of S that it gives us.

- **Every-visit Monte-Carlo Policy Evalutation:**
  - To evaluation state s
  - *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 $V(s) = \frac{S(s)}{N(s)}$
  - Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$
- **Incremental Mean**
  - The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally
  - $\mu_k = \frac{1}{k} \sum_{j=1}^{k} x_j$
  - $\mu_k = \frac{1}{k}(x_k + \sum_{j=1}^{k-1} x_j)$
  - $\mu_k = \frac{1}{k}(x_k + (k-1)\mu_{k-1})$
  - $\mu_k = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$
  - This final equation can be thought of as taking the old mean μ_{k-1} and taking a small step with size $\frac{1}{k}$ towards the value we just saw $(x_k - \mu_{k-1})$
- **Incremental Monte-Carlo Updates:**
  - Update $V(s)$ incrementally after each episode $S_1, A_1, R_2, \ldots, S_T$
  - For each state S_t with return G_t:
  - $N(S_t) \leftarrow N(S_t) + 1$
  - $V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$
  - In non-stationary problems it can be useful to track a running mean, i.e. to forget old episodes: $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
- **Temporal-Difference Learning Overview:**
  - TD methods learn directly from episodes of experience
  - TD is *model-free*: no knowledge of MDP transitions or rewards
  - TD learns from *incomplete* episodes, by bootstrapping This is in contrast to MC which must use full episodes.
  - Instead of taking complete episodes to determine rewards, TD can take partial episodes and then use estimates to guess what the reward for the rest of the episode would be.
  - TD updates a guess towards a guess

- **MC and TD:**
  - Goal: learn v_π online from experience under policy π
  - Incremental every-visit Monte-Carlo
    - Update value $V(S_t)$ toward actual return G_t: $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
  - Simplest temporal-difference learning algorithm: TD(0)
    - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$:
    - $V(S_t) \leftarrow V(S_t) + \alpha((R_{t+1} + \gamma V(S_{t+1}) - V(S_t)))$
    - $R_{t+1} + \gamma V(S_{t+1})$ is called the TD *target*
    - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD *error*
  - At each step of TD you are updating your estimate of what you thought would happen with what did happen.
  - With MC you update each step by correcting based on what happened with the entire episode rather than with just that step.
- **Advantages and Disadvantages of MC vs. TD:**
  - TD can learn *before* knowing the final outcome
    - TD can learn online every step
    - MC must wait until end of episode before return is known
  - TD can learn *without* the final outcome
    - TD can learn from incomplete sequences
    - MC can only learn from complete sequences
    - TD works in continuing (non-terminating) environments
    - MC only works for episodic (terminating) environments
- **Bias/Variance Trade-Off:**
  - Return $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
  - True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
  - TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
  - TD target is much lower variance than the return:
    - Return depends on many random actions, transitions, rewards
    - TD target depends on one random action, transition, reward
    - In other words, we are only subject to the noise from the single step with the TD target vs. from the entire trajectory with the return
- **Continued Advantages and Disadvantages of MC vs. TD:**

- MC has high variance and zero bias
  - Good convergence properties
  - (even with function approximation)
  - Not very sensitive to initial value
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - TD(0) converges to $v_\pi(s)$
  - (but not always with function approximation)
  - More sensitive to initial value
- **Certainty Equivalence:**
  - MC converges to solution that minimizes mean-squared-error
  - This is the best fit to all observed returns
  - TD(0) converges to solution of MDP that best explains the data
  - Think of this as first fitting for an MDP and then solving for the MDP
- TD exploits the Markov property - is usually more efficient in Markov environments
- MC does not exploit Markov property and is usually more efficient in non-Markov environments
- **Bootstrapping:**
  - Update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling:**
  - Update samples an expectation
  - MC samples
  - DP does not sample
  - TD does sample
- **TD(λ):**
  - Let TD look n steps into the future before deciding how to update
  - $n = \infty$ is Monte-Carlo
  - Recall, TD (n = 1) has update $V(S_t) \leftarrow V(S_t) + \alpha((R_{t+1} + \gamma V(S_{t+1})) - V(S_t))$
  - n = 2: $V(S_t) \leftarrow V(S_t) + \alpha((R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+3}) - V(S_t))$
  - Define $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$

    * Then n-step TD learning updates with $V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$
    * We can average n-step returns over different n
    * e.g. average 2-step and 4-step returns: $\frac{1}{2}G^{(2)} + \frac{1}{2}G^{(4)}$
    * This combines information from two different time-steps
    * Doing this across all n gives λ-return which is like a geometrically weighted average across all n
    * $G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$
    * λ is giving the weight for each successive n
    * The weighting for a given n is $(1-\lambda)\lambda^{n-1}$
    * Now use this for TD learning update: $V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\lambda} - V(S_t))$
  - **Eligibility Traces:**
    * Combine frequency and recency heuristics to solve problem of assigning credit to the right factor
    * $E_0(s) = 0$, $E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$
  - **Backward View TD(λ):**
    * Keep eligibility trace for every state s
    * Update value $V(s)$ for every state s
    * In proportion to TD-error δ_t and eligibility trace $E_t(s)$
    * $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
    * $V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$
  - **TD(λ) and TD(0):**
    * When $\lambda = 0$, only current state is updated
      $E_t(s) = \mathbf{1}(S_t = s)$
      $V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$
    * This is exactly equivalent to TD(0) update