

CS 581 – ADVANCED ARTIFICIAL INTELLIGENCE

TOPIC: REINFORCEMENT LEARNING



Mustafa Bilgic



<http://www.cs.iit.edu/~mbilgic>



<https://twitter.com/bilgicm>

SOURCES

- Reinforcement Learning – an Introduction
 - by Sutton & Barto
 - <http://incompleteideas.net/book/the-book-2nd.html>
- Introduction to RL lectures
 - by DeepMind; David Silver
 - <https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver>
 - <https://www.youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ>

REQUIRED BACKGROUND

○ MDPs

- Value function U , action-value function Q , policy π
- Bellman equations
- Value iteration
- Policy iteration

○ Multi-armed bandits

- Exploration vs exploitation trade-off
- ϵ -greedy approach

RL AND MDPs

- MDPs are building blocks for RL
- RL has the additional complexity that the agent does not have access to the full specification of the MDP. For e.g.,
 - Transition probabilities are often unknown
 - Reward function is often unknown

PREDICTION AND CONTROL

- Prediction
 - Given a policy, estimate the value function
- Control
 - Find the optimal policy

MODEL-FREE VS MODEL-BASED

- Model-free:
 - The agent does not have and does not learn a model of the how the environment works
- Model-based:
 - The agent learns/improves a model of the environment
- Note: we are not talking about an approximate “model” of a state representation (such as DL for value estimations of states); rather, we mean model of the environment, such as transition probabilities

OUTLINE

- Prediction
 - Monte Carlo methods
 - Temporal-difference learning
- Control
 - Monte Carlo methods
 - Temporal-difference learning
 - Sarsa

Monte Carlo Prediction

- Task: prediction
 - Policy is given; compute the value functions
- Unknown environment dynamics
 - If the transition probabilities and the reward function were given, we could use the algorithms we saw earlier for complete MDPs
- Learn from *experience* – samples of episodes
- *Episode* is a sequence of state, action, reward triplets
- Assume all episodes reach a terminal state
- Basic idea: expectation = average over samples

MC PREDICTION – PSEUDOCODE

- Given a policy π
- Initialize $V(s)$ for all s
- Loop:
 - Sample an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$
 - For each state s in the episode, compute the cumulative discounted reward starting from that state s
- $V(s)$ is the average cumulative discounted reward starting from state s
- If s appears multiple times in a single episode (i.e., loops):
 - First-visit MC: for each episode, only the first appearance of the state s is considered
 - Every-visit MC: every appearance is considered and averaged accordingly
- Converges to the true values as the number of visits approach infinity

MC PREDICTION – PSEUDOCODE

- Given a policy π
- Initialize $Q(s, a)$ for all (s, a) pairs
- Loop:
 - Sample an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$
 - For each state (s, a) pair in the episode, compute the cumulative discounted reward starting from that state s and taking action a
- $Q(s, a)$ is the average cumulative discounted reward starting from state s and taking action a
- Advantage of estimating Q instead of V :
 - Can use Q values to find a better policy, i.e., for control
 - Caveat: we need to explore other actions to find a better one (remember the exploration vs exploitation trade-off)

INCREMENTAL UPDATE

- Task: update the mean over a sequence at each step
- Assume we have seen x_1, x_2, \dots, x_n
- $\mu_n = \frac{1}{n} \sum_{i=1}^n x_i$
- We receive x_{n+1}
- $\mu_{n+1} = \frac{1}{n+1} \sum_{i=1}^{n+1} x_i$
- $\mu_{n+1} = \frac{1}{n+1} (x_{n+1} + \sum_{i=1}^n x_i)$
- $\mu_{n+1} = \frac{1}{n+1} \left(x_{n+1} + \frac{n}{n} \sum_{i=1}^n x_i \right)$
- $\mu_{n+1} = \frac{1}{n+1} (x_{n+1} + n\mu_n)$
- $\mu_{n+1} = \frac{1}{n+1} (x_{n+1} + (n+1)\mu_n - \mu_n)$
- $\mu_{n+1} = \mu_n + \frac{1}{n+1} (x_{n+1} - \mu_n)$
- Next estimate = current estimate + learning rate * error

INCREMENTAL UPDATE

- $\mu_{n+1} = \mu_n + \frac{1}{n+1} (x_{n+1} - \mu_n)$
- $\mu_{n+1} = \mu_n + \alpha (x_{n+1} - \mu_n)$
- When can incrementally update the MC means for the value (or action-value) functions after each episode
- Using a fixed α instead of $\frac{1}{n+1}$ allows us to handle non-stationary cases where the dynamics of the system changes
 - Recent experiences count more than earlier ones

TEMPORAL DIFFERENCE

- In MC, each episode had to end at a terminal state
- MC ignored the Bellman equations
- Can we learn from partial/incomplete episodes?
 - Yes, but we need estimates of the values
- The method we will use is called Temporal Difference (TD) learning

TD PREDICTION

- Given a policy π
- For an episode:
 - MC
 - $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
 - Need to wait till the end to calculate G_t
 - TD: take one step and use Bellman equation
 - $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- This is called TD(0) because it updates values based on a single look ahead

TD(0) PREDICTION

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 S

 Loop for each step of episode:

$A \leftarrow$ action given by π for S

 Take action A , observe R, S'

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

$S \leftarrow S'$

 until S is terminal

Figure from <http://incompleteideas.net/book/the-book-2nd.html>

MC CONTROL

- Unlike prediction, we are not given a policy
- The environment dynamics are unknown
- The agent needs to find an optimal policy through experience – samples of episodes
- Can't we combine policy iteration of MDPs and MC sampling?
 - Yes, but
 - We need to learn a Q function; V function would not be useful because we do not have the transition function to read the policy from V values
 - We cannot sample from a deterministic policy; otherwise, the agent does not learn
 - We need to balance exploration vs exploitation
 - We will use ϵ -greedy approach
 - During sampling an episode:
 - With ϵ probability, choose a random action
 - Otherwise, choose the action recommended by the policy

MC CONTROL – PSEUDOCODE

- Initialize $Q(s, a)$ for all (s, a) pairs
- Initialize a policy π
- Loop:
 - Sample an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$
 - For choosing an action A_i at state S_i :
 - With ϵ probability, choose a random action
 - Otherwise, choose the action recommended by the current policy
 - For each state (s, a) pair in the episode, compute the cumulative discounted reward starting from that state s and taking action a
 - Update Q using the sample averages
 - Update π using the updated Q

TD CONTROL – SARSA

- Like MC Control, except we use the one step look ahead instead of waiting till the episode terminates
- $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
- Because we use $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$, this algorithm is also called SARSA
- Like MC Control, we need to introduce randomness into choosing actions, rather than strictly following a deterministic policy
 - Use ϵ -greedy approach

SARSA ALGORITHM

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

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)

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

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

 until S is terminal

Figure from <http://incompleteideas.net/book/the-book-2nd.html>