MC and TD Learning

Whats New?

- We want to estimate the value function without having access to the dynamics model
- We want to interact with the environment and collect experiences
- We average sampled returns to calculate the expectations (earlier we used the known probability distribution)

Prediction Problem vs Control Problem

- **Prediction problem** : Policy evaluation, calculating the value function of a given policy pi (eg. Monte Carlo Prediction)
- **Control problem** : General policy iteration (GPI), getting the optimal policy (eg. Monte Carlo Control)

Monte Carlo Methods

- All episodes must terminate (learning can only happen from complete trajectories)
- Only defined for episodic tasks
- Basic idea : Vpi(s) = mean return

Monte Carlo Prediction / Monte Carlo Policy Evaluation

- Goal : Learn Vpi(s) using episodes under policy pi
- Return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Value function using the expected return

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

- We use the empirical mean return to estimate the expected return

Every Visit vs First Visit MC

- **Every visit** : while averaging, consider returns for every time state s is visited in a trajectory
- **First visit** : while averaging, consider returns only for the first time strate s is visited in a trajectory

Monte Carlo Control

- Monte Carlo Prediction / Monte Carlo Policy Evaluation
- Policy Improvement Step : gredify with respect to the value or action value function

Exploration Problem

- Dilemma : Policies need to act sub optimally in order to explore all actions

- Solutions :
    - **Exploring Starts** : every state action pair has non-zero probability of being the start
    - **Epsilon Soft Policies** : Policy never becomes deterministic, epsilon is always non-zero
    - **Off Policy** : Use different policy to collect experiences (behavior policy) and then update the target policy
        - Importance Sampling : Weight returns by the ratio of probabilities of the same trajectory under the two policies

Temporal Difference (TD) Learning

- General equation:

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- TD(0) / One Step TD :

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

TD Target : $R_{t+1} + \gamma V(S_{t+1})$

TD Error : $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

TD Prediction / TD Policy Evaluation

- Goal : Find Vpi(s) using episodes from policy pi
- TD(0) uses:

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$

Advantages of TD Learning

- Like MC, does not need to know the model dynamics
- Bootstrap : learning a guess from a guess
- Does not need to end for the end of an episodes (some episodes can be really long!)
- Requires lesser memory and computation
- In practice, converges faster than MC methods

Bias Variance Trade-Off between MC and TD

- MC has zero bias, high variance
- TD has low variance, some bias

SARSA and Q-Learning

- SARSA is On-Policy TD Control
- Q-Learning is Off-Policy TD Control

MCTS

Online Planning

- Concept of ‘mental unrolling’
- Unroll the model of the environment forward in time to select the right action sequences to achieve your goal

- Since we have the model, why not learn the value function of every state directly?
- Because there can be extremely large number of possible states

Limitation : Cannot Exhaustively Search

- Curse of dimensionality : not possible to search the entire tree

- Solution :
    - **Reduce Depth** by truncating the search stree at a state s and then use the approximate value of s
    - **Reduce Breadth** by sampling actions from a policy pi(a|s) which is probability distribution of actions given state s

Internal and External Policies

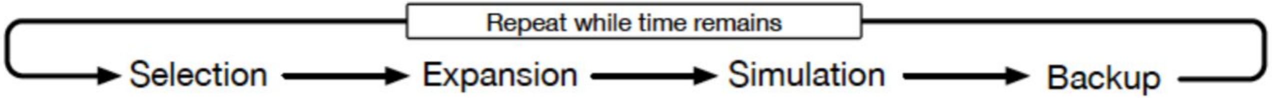
- **Internal Policy** : Keep track of action values for the root and nodes internal to the tree (which is being expanded), and use these to improve the simulation policy over time
- **External Policy** : We donot have Q estimates, therefore we use a random policy

Upper Confidence Bound

$$A_t = \operatorname{argmax}_a \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$$

- Probability of choosing an action:
    - Increases with a node’s value : Qt(a) (exploitation)
    - Decreases with number of visits : Nt(a) (exploration)

The 4 Steps



- Selection :
  - For nodes we have seen before
  - Pick actions according to UCB
- Expansion :
  - Used when we reach the frontier
  - Add one node per rollout
- Simulation :
  - Used beyond the frontier
  - Pick actions randomly
- Back-Propagation :
  - After reaching a terminal node
  - Update values and visits for selected and expanded states