Section overview

Model-Free Learning

Model-Free Learning

- Past: Planning by dynamic programming
  Solve a known MDP (no learning), but we learned to optimise our planning using DP.
- Now: Model-free prediction ("Policy evaluation")
  Estimate the value function of an unknown MDP
- Next: Model-free control ("Policy improvement")
  Optimise the value function of an unknown MDP

Model-Free Reinforcement Learning: Monte Carlo (MC)

1. MC methods learn directly from episodes of experience
2. MC is model-free: no knowledge of MDP transitions or rewards needed
3. MC learns from complete episodes (of sample traces): no bootstrapping
4. MC uses the simplest possible idea: value of state = mean return
   $\Rightarrow$ BUT this can only be applied to episodic MDPs that have terminal states.

Monte Carlo (MC) Methods

We want to learn the value function for a given policy π.

- Recall that the value of a state is the expected return–expected cumulative future discounted reward–starting from that state.
- An obvious way to estimate it from experience, then, is simply to average the returns observed after visits to that state.
- As more returns are observed, the average should converge to the expected value.

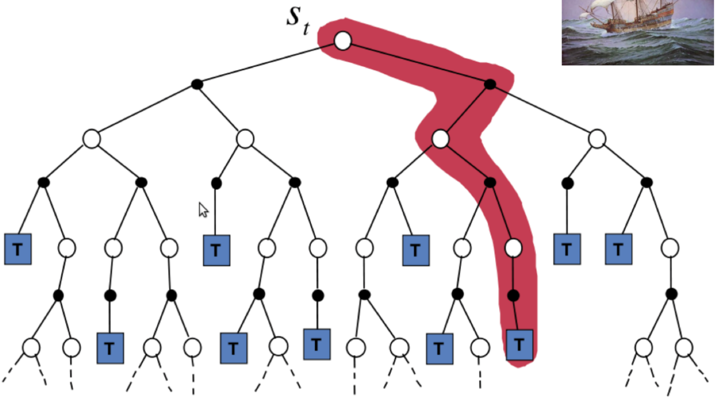
This simple idea underlies all Monte Carlo methods.

MC Policy Evaluation

- Goal: learn V^π from traces τ of episodes of length T that we experience under policy π
  $\tau \equiv s_1, a_1, r_2, \ldots, s_k$
- We already defined the return as the total discounted reward:
  $R_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_T$
- We already defined the value function as the expected return:
  $V^\pi(s) = \mathbb{E}\left[R_t | S_t = s\right]$
- Monte-Carlo policy evaluation uses empirical mean returns instead of expected return.

Monte Carlo performs sample trace evaluations

Another way is to sample values for V



Monte-Carlo Policy Evaluation

procedure MonteCarloEstimation(π)
  Init
    $\hat{V}(s) \leftarrow$ arbitrary value, for all $s \in S$.
    $Returns(s) \leftarrow$ an empty list, for all $s \in S$.
  EndInit
  repeat
    Get trace, τ, using π.
    for all s appearing in τ **do**
      $R \leftarrow$ return from first appearance of s in τ.
      Append R to $Returns(s)$
      $\hat{V}(s) \leftarrow$ average($Returns(s)$)
  until forever

Above we have the First visit MC algorithm, as we append the return of the episode from the first occurrence of a state s. Another version is Every visit MC, where we append the return of the episode (from that point) on every occurrence of state s in the episode.

Example: First visit vs Every visit

For a single entry, the $Returns(s)$ list of returns is averaged over either the first or all entries of an episode. To simplify the problem so we can write it u, we assume a $\gamma = 0$ (so return and immediate reward are the same)

Trace example:

(S_0, a=E,r=10) (S_1,a=E, r=-10) (S_0 ,a=E, r=10)
(S_3 , a=E, r=20) (S_1 , a=E, r=0) (S_4 , a=E, r=100)

Every-visit MC

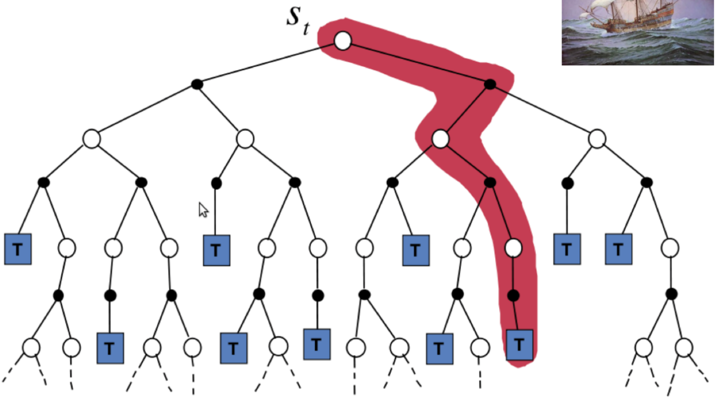
S,a=E	
S_0	10,10
S_1	-10,0
S_2	
S_3	20
S_4	100

First-visit MC

S,a=E	
S_0	10, 10
S_1	-10,0
S_2	
S_3	20
S_4	100

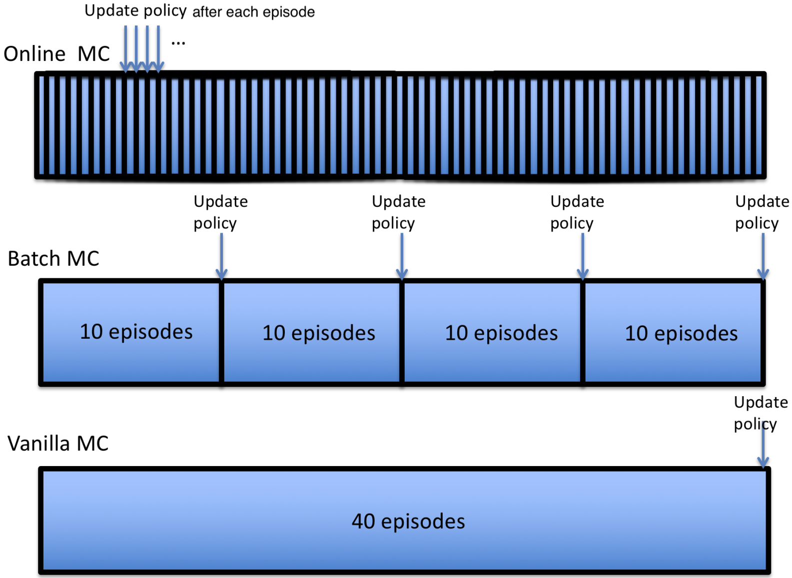
Monte Carlo performs sample trace evaluations

Another way is to sample values for V



But, remember "you cannot backup death"
(what does this imply?)

Batch & Online Monte-Carlo



Batch vs online averaging

Normally we batch process data $x_1, x_2, \ldots$ to calculate its mean μ

$$\mu = \frac{1}{k}\sum_{j=1}^{k} x_j \tag{35}$$

In an RL world we would like to be able to do this online while we experience new data and compute the current means $\mu_1, \mu_2, \ldots$ efficiently.

$$\begin{aligned} \mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\ &= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\ &= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\ &= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right) \end{aligned}$$

Incremental estimation updates

Consider the structure of the update on the average

$$\mu_k = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

It has the form of an incremental estimation computation

$$\Delta = \mu_k - \mu_{k-1} = \frac{1}{k}(x_k - \mu_{k-1})$$

- a small weighting factor ≤ 1
- an old estimated value μ that is updated with
- new data x ("the difference pulls the estimate in the direction of the data")

We will encounter this structure difference structure throughout Model-Free Learning.

Incremental Monte-Carlo Updates

We can now update value functions without having to store sample traces:

1. Update $V(s)$ incrementally after step $s_t, a_t, r_{t+1}, s_{t+1}$
2. For each state s_t with return R_t (up to this point) and $N(s)$ the visit counter to this state:

$$\begin{aligned} N(s_t) &\leftarrow N(s_t) + 1 \\ V(s_t) &\leftarrow V(s_t) + \frac{1}{N(s_t)}(R_t - V(s_t)) \end{aligned}$$

Moreover, if the world is non-stationary, it can be useful to track a running mean, i.e. by gradually forgetting old episodes.

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t))$$

The parameter α controls the rate of forgetting old episodes (**learning rate**).
Why should we consider non-stationary conditions?