5.0 : Intro

First learning method for estimating value functions and discovering optimal policies. Here, we don't have knowledge of the environment dynamics, we learn only by experience.

Monte Carlo methods are based on episodes and averaging sample returns. They are incremental in the episode by episode sense (not in a step by step sense).

As in dynamic programming, we adapt the idea of general policy iteration. Instead of computing the value function from our knowledge of the MDP, we learn it from sample returns.

5.1 : Monte Carlo Prediction

Here, we focus on learning the state-value function for a given policy π. Recall: the value of a state is the expected return - the expected cumulative future discounted reward - starting from that state.

An obvious way to do it is to estimate from experience, by averaging the returns observed after visits to that state. As more returns are observed, this value should converge to the expected value.

- we wish to estimate $v_\pi(s)$, the value of state s under policy π.
- we have a set of episodes using π and passing through s.
- The first visit of s is the first time s is visited in an episode
- First visit MC method: averages the returns following first visit to s in each episode, from all episodes.
- Every visit MC method: averages the returns of all visit to s, from all episodes.

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$ //initialize $V(s)$ randomly.

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$: //works backward from terminal.

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

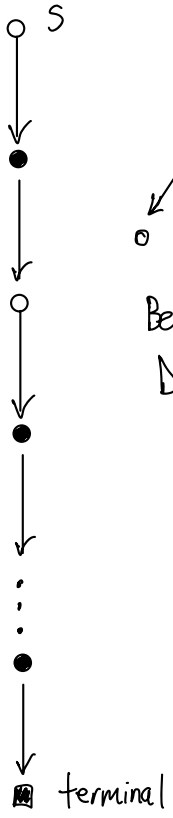
$V(S_t) \leftarrow$ average$(Returns(S_t))$

For every-visit version, it's the same without the check for S_t having occurred earlier in the episode.

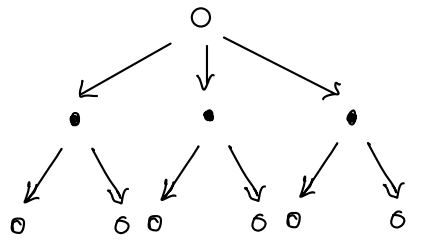
How to apply back-up diagrams to MC methods:

Go all the way to the end of the episode, and show only the sampled transitions instead of all possible transitions.

The estimate for each state in MC method is independent from estimates of other states. In other words, MC methods do not bootstrap.



MC estimation of V_π



Bellman Eq for V_π, Dynamic programming.

MC:
- only sampled transitions
- All the way to the end of episode
- no bootstrapping
- high variance, but unbiased.
- not incremental per timestep, but incremental per episode.

DP:
- All possible transitions
- only 1 step
- bootstrapping
- lower variance

sample of return:

$$G_{s_t} = \sum_{i=t}^{T-1} r_i \Rightarrow V(s) = E[G_{s_t}]$$

the return of state s if starting at time t.

sum of all rewards along the way (undiscounted version).

5.2: Monte Carlo methods for Action Values

If the model of a system is not available, it's better to estimate action values rather than state values.

That's because if we only have the state values we need to do a one-step lookahead that needs the model (p here):

$$\pi(s) = \arg\max_a \sum_{s', r} p(s', r \mid s, a)[r + \gamma V_\pi(s')]$$

MC methods are primarily used to find an estimate of q_*, since if we have the q-value we don't need the model p to find the policy.

Policy Evaluation

- Every visit MC: estimates the value of $q_\pi(s, a)$ as the average of the returns that have followed all the visits to a state-action node.
- First visit MC: Same but taking only the first time the node has been visited in each episode.

Problem: many state-action pairs may not be visited after all. (For example, if we have a deterministic policy.) We must assure continual exploration:

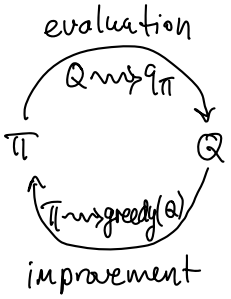
- exploring starts; one way to do it is to start each episode in a state-action pair, each having a non-zero probability of being selected.
  - Practically not feasible; can't just randomly start in a state.
- use a stochastic policy with non-zero probability of selecting each action.

5.3: Monte Carlo Control

This section is about how MC estimation can be used in control (aka approximate optimal policies).

To begin, we consider the MC version of classical policy iteration. We perform alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy π_0 and ending with the optimal policy and optimal value function.

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_{\pi_*}$$



// policy iteration.

Policy evaluation: Many episodes are experienced, with the approximate action-value function approaching q_*, asymptomtically. We assume for now that we observe an infinite amount of episodes generated with exploring starts.

Policy Improvement is done by making the current policy greedy wrt the current action-value function:

$$\pi(s) \doteq \arg\max_a q(s,a)$$

policy = choose action w/ highest reward.

Here, we construct π_{k+1} as the greedy policy wrt $q_\pi(s,a)$. The policy improvement theorem applies to π_k and π_{k+1} because $\forall\ s \in S$:

$$q_{\pi_k}(s, \pi_{k+1}) = q_{\pi_k}(s, \operatorname{argmax}_a q_{\pi_k}(s, a))$$
$$= \max_a q_{\pi_k}(s, a)$$
$$\geq q_{\pi_k}(s, \pi_k(s))$$
$$\geq v_{\pi_k}(s), \quad \forall s \in S$$

This property assures us that each π_{k+1} is uniformly better than π_k and that in turn ensures that the overall process converges to optimal policy and optimal state-action function.

Assumptions we made:

- episodes have exploring start (need this for control; if all states are not visited then we can't develop an optimal policy).
- policy evaluation can be done with an infinite # of episodes.

To obtain a pratical algorithm we will need to remove both assumptions. The second one is easy to remove, since the same issue arises in dynamic programming (DP) problems. In both DP and MC, there are two ways of solving the problem:

- iterate policy evaluation until a threshold of convergence has been reached.
- only make a fixed number of steps for policy evaluation (1 in value iteration).

For MC methods it's natural to alternate btwn evaluation and improvement on a per-episode basis; After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited during the episode.

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ // Q, not V.

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$: // fill in from end to start.

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$ //greedy

All the returns for state-action pairs are averaged, no matter what policy was in force when they were observed. Such a Monte-Carlo algorithm cannot converge to any suboptimal policy.

If it did, then the value function would eventually converge to the value function for that policy, and that would in turn make the policy change. Stability is achieved only when both policy and value function are optimal.

5.4: Monte Carlo control without Exploring Starts

we want to ensure that all actions are selected infinitely often.

- on-policy methods: evaluate and improve the policy that is being used to make decisions (such as the monte Carlo with Exploring Starts method above).
- Off-policy methods: evaluate and improve a policy that is different than the one used to generate the data.

In this section we focus on the on-policy part, and then the off-policy will be right after.

On-policy control methods usually use soft policies, meaning that $\pi(s,a) > 0$ for all a and s, but gradually shifted closer and closer to a deterministic optimal policy. Here we use an ε-greedy policy, giving $\frac{\varepsilon}{|A(s)|}$ to all non-greedy actions and $1 - \varepsilon + \frac{\varepsilon}{|A(s)|}$ to the greedy actions. That means the Greedy action is selected most of time, but sometimes (with probability ε) we select an action at random.

The overall idea of on-policy Monte Carlo control is still that of General Policy Improvement (GPI).

1) Policy Evaluation: use first-visit MC to estimate the action value for current policy.

2) Policy Improvement: we can't just make the policy greedy wrt the current action-values because it would prevent exploration of non-greedy actions. Fortunately, GPI does not require that the policy be taken all the way to a greedy policy, only that it moved towards a greedy policy.

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

} initialize π and $Q(s,a)$ randomly.

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$: // end to start

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ // greedy action (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$ // ε-greedy policy.

5.5: Off-Policy Prediction by Importance Sampling

In the off-policy setting, we use two policies:

- The target policy is the policy being learned; "Goal" policy. π
- The behavior policy is the policy used to generate actions and collect data during training. b

Off-policy methods are usually a bit harder bc they require additional notation, and they often have higher variance (the data is from a different policy), and are slower to converge. On the other hand, they are usually more powerful.

we begin the study of off-policy methods by considering the prediction problem, in which both target and behavior policies are fixed. That is, we wish to estimate v_π or q_π but all we have is a bunch of episodes generated by $b \neq \pi$. we also require that all actions taken by π can be taken by b: $\pi(a|s) > 0$ implies $b(a|s) > 0$. (coverage assumption).

The target policy π may be deterministic, often in control it's the deterministic greedy policy wrt the current estimate of the value function. This policy becomes a deterministic policy while b remains stochastic and more exploratory. For now, in the prediction problem, we consider that π is given and unchanged.

Importance Sampling :

IS is a technique for estimating expected values under one distribution given samples from another, which is our situation here.

We will weight the returns according to the relative probability of their trajectories occurring under the target and behavior policies, called the importance-sampling ratio.

Given a starting state S_t, the probability of subsequent state-action trajectory $A_t, S_{t+1}, \dots, S_T$ under any policy π is:

$$\Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim \pi\}$$

$$= \pi(A_t, S_t)\, p(S_{t+1} \mid S_t, A_t)\, \pi(A_{t+1} \mid S_{t+1}) \dots p(S_T \mid S_{T-1}, A_{T-1})$$

$$= \boxed{\prod_{k=t}^{T-1} \pi(A_k, S_k)\, p(S_{k+1} \mid S_k, A_k)}$$

Thus, the importance sampling ratio $\rho_{t:T-1}$ of the trajectory from time t to T (we stop the ratio at $T-1$ because we don't have the A_T action) is:

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k, S_k)\, p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k, S_k)\, p(S_{k+1} \mid S_k, A_k)} = \boxed{\prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}}$$

The probabilities p that depends on the MDP nicely cancel out and the importance sampling ratio thus depends only on the two policies involved.

Estimating V_π

we still wish to estimate the expected return under the target policy π but we only have access to the returns G_t of the behavior policy b. These returns have the 'wrong' expectation $E[G_t \mid S_t = s] = V_b(s)$ and cannot be averaged to obtain $V_\pi(s)$. This is where the importance sampling ratio comes in:

$$V_\pi(s) = E\left[\left(\rho_{t:T-1}\right)\left(G_t\right) \mid S_t = s\right]$$

Now we are ready to estimate $V_\pi(s)$ using returns from a batch of observed episodes following b

notation: if episode n finishes at time step $t = 100$, episode $n+1$ starts at $t = 101$. We can then define $\mathcal{T}(s)$, the set of all time steps in every episode where state s has been visited.

For a first-visit method, $\mathcal{T}(s)$ would only include time steps of the first visit to s in each episode.

- $T(t)$ denotes the first time of termination after timestep t.
- $G(t)$ denotes the return after the t to $T(t)$ trajectory.

To estimate $V_\pi(s)$ we simply scale the returns by the ratios and average the results:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \left\{\left(\rho_{t:T(t)-1}\right)\left(G_t\right)\right\}}{|\mathcal{T}(s)|}$$

This is the ordinary importance sampling, because we use a simple average.

we can use a weighted average and get the weighted importance sampling:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \left(\rho_{t:T(t)-1}\right)\left(G_t\right)}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}} \qquad Q(s,a) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \left(\rho_{t+1:T(t)-1}\right)\left(G_t\right)}{\sum_{t \in \mathcal{T}(s)} \rho_{t+1:T(t)-1}}$$

To understand the difference btwn the two, consider the estimates after observing a single return.

- For the weighted average, the ratio cancels out and the estimate is equal to the return observed. It's a reasonable estimate but its expectation is $V_b(s)$ rather than $V_\pi(s)$, so it is biased.
- for the ordinary importance sampling, the expectation is $V_\pi(s)$ so it's not biased but it can be a bit extreme. If the trajectory is 10 times more likely under π than under b, the estimate would be 10 times the observed return, which would be quite far from the actually observed return.

Infinite Variance

More formally we can express the differences btwn the two estimates (ordinary and weighted) in terms of bias and variance;

- bias: the ordinary is not biased but the weighted is.
- variance: is generally unbounded for the ordinary estimate because the variance of the ratios can be unbounded whereas in the weighted estimator the largest weight on any single return is 1. The weighted estimator

has dramatically lower variance in pratice and thus is strongly preferred.

Example of variance for ordinary importance sampling when the trajectory may contain loops:

we can verify that the variance of the importance-sampling scaled returns is infinite in this last example. The variance of a random variable X is the value of the deviation from its mean $\bar{X}$.

$$\text{Var}[X] \doteq E\left[(X-\bar{X})^2\right] = E\left[X^2 - 2X\bar{X} + \bar{X}^2\right] = E\left[X^2\right] - \bar{X}^2$$

If the mean of the random variable is finite (as in this example), having an infinite variance means that the expectation of the square of the r.v. is infinite. Thus, we only need to show that the expected square of the importance-sampling scaled return is infinite:

$$E_b\left[\left(\prod_{t=0}^{T-1} \frac{\pi(A_t|S_t)}{b(A_t|S_t)} G_0\right)^2\right]$$

To compute this expectation, we need to break it down into cases based on episode length and termination.

- Any episode ending with right action can be ignored bc the target policy would never taken this action.

- we need only consider episodes that involve some number (possibly zero) of left actions that transition back to the non-terminal state, followed by a left action transitioning to termination.
- G_0 can be ignored because all considered episodes have a return of 1.
- we need to consider each length of episode, multiplying the probability of the episode's occurence by the square of its importance-sampling ratio, and add them up:

5.6: Incremental Implementation

We want to implement those MC prediction methods on a episode-by-episode basis.

To do this, we will take inspiration from Ch 2, where we incrementally computed Q estimates.

For the on-policy method, the only difference is that here we average returns whereas in Ch 2 we averaged rewards.

For off-policy, we need to distinguish btwn ordinary importance sampling and weighted importance sampling.

In ordinary importance sampling, the returns are scaled by the importance sampling ratio $\rho_{t:T(t)-1}$ (eq 5.4) then simply averaged like in eq 5.6. For these methods we can again use the incremental methods of ch 2, but using the scaled returns instead of the reward.

For weighted importance sampling we have to form a weighted average of the returns, and use a slightly different incremental algorithm.

Suppose we have a sequence of returns $G_1, G_2, \dots, G_{n-1}$, all starting in the same state and each with a corresponding random weight W_i (for example $\rho_{t:T(t)-1}$). We wish to form the estimate (for $n \geq 2$):

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}$$

and keep it up to date as we obtain a single additional return G_n. In addition to keeping track of V_n, we must maintain for each state the cumulative sum C_n of the weights given to the first n returns. The update rule for V_n ($n \geq 1$) is:

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\left[G_n - V_n\right]$$

and

$$C_{n+1} \doteq C_n + W_{n+1}$$

complete algorithm below (extends on the on-policy case by setting $\pi = b$):

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}\left[G - Q(S_t, A_t)\right]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

5.7: Off-Policy Monte-Carlo Control

In this book we consider two classes of learning control methods:

- on-policy methods: estimate the value of a policy while using it for control (finding the optimal policy).
- off-policy methods: these two functions are separated (target vs behavior policies).

Off-policy MC control follow the behavior policy while learning about improving the target policy.

The algorithm below shows the off policy MC method based on GPI and importance sampling, for estimating π_* and q_*.

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \in \mathbb{R}$ (arbitrarily)
  $C(s,a) \leftarrow 0$
  $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
  $b \leftarrow$ any soft policy
  Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$ (with ties broken consistently)
    If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$

A potential problem is that this method only learns from the tails of episodes, when all of the remaining actions are greedy. If non-greedy actions are common, then learning will be slow, particularly for states appearing in early portions of long episodes.

Maybe incorporating TD-learning can help. Also if $\gamma < 1$, the idea developed in the next chapter can help.

5.7b: Safe off-policy evaluation

Return probabilistic lower bound V_π^{lb} such that:
$V_\pi \geq V_\pi^{lb}$ with prob. $1-\delta$ Given: π, δ, data from π_b without ever running policy π!

Confidence bounds: Chernoff-Hoeffding inequality
with probability at least $1-\delta$:

$$\mu \geq \frac{1}{n}\sum_{i=1}^{n} x_i - b\sqrt{\frac{\log(1/\delta)}{2n}} \quad \text{for} \quad 0 \leq x_i \leq b$$

$$\downarrow$$

$$V_\pi \geq \frac{1}{n}\sum_{i=1}^{n} G_i^{\pi_b} \cdot \rho_i^{\pi,\pi_b} - G_{max}\sqrt{\frac{\log(1/\delta)}{2n}} \quad \text{for} \quad 0 \leq G_i \leq G_{max}$$

5.8 : * Discounting-Aware Importance Sampling

5.9 : *Per-decision Importance Sampling

Given returns $G_1, \dots, G_n$ and $\rho_i = \prod_{k=t_i}^{\tau_i - 1} \frac{\pi(A_k \mid S_k)}{\pi_b(A_k \mid S_k)}$

Then $V_a(s) =$

OIS:

$$\frac{1}{n} \sum_{i=1}^{n} G_i \rho_i$$

WIS:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{G_i \rho_i}{\sum_{j=1}^{n} \rho_i}$$

PDIS:

$\frac{1}{n} \sum \tilde{G}_i$, where:

$\tilde{G}_i = \rho_{1:1} R_1 + \gamma \rho_{1:2} R_2 + \dots + \gamma^{n-1} \rho_{1:t} R_t$

and $\rho_{a:b} = \prod_{k=a}^{b} \frac{\pi(A_k \mid S_k)}{\pi_b(A_k \mid S_k)}$

5.10: Summary

Monte-Carlo methods learn value functions and optimal policies from experience in the form of sample episodes.

This has advantages over DP methods:

- we don't need a model
- we can use simulation or sample models
- It's easy to focus on a small subset of states.
- They don't update their value estimate for a state based on estimates of successor states (no boot strapping) thus they can be less harmed by violations of the Markov property.

We still use GPI by mixing policy evaluation and policy improvement on a episode-by-episode basis, and for the policy evaluation part we simply average the returns for a given state instead of using the model to compute value for each state.

Exploration is an issue, that we can address with:

- exploring starts
- off-policy methods

off-policy methods are based on importance sampling, that weigh the return by the ratio of the probabilities of taking the observed actions under the two policies, thereby transforming their expectations from the behavior policy to the target policy.

ordinary importance sampling uses a simple average of the weighted returns, whereas weighted importance sampling uses a weighted average.

In the next chapter, we consider methods that make use of experience (like MC) but do bootstrapping (like DP).

$G_i^{s,\pi}$: i^{th} return starting from state s, collected from policy π.

on-policy prediction:

$$V_\pi(s) = \frac{1}{N}\sum_{i=1}^{N} G_i^{s,\pi}$$

$$Q_\pi(s,a) = \frac{1}{N}\sum_{i=1}^{N} G_i^{s,a,\pi}$$

off-policy prediction:

$$V_{\pi'}(s) = \frac{1}{N}\sum_{i=1}^{N} G_i^{s,\pi} \cdot e_i$$

$$Q_{\pi'}(s,a) = \frac{1}{N}\sum_{i=1}^{N} G_i^{s,a,\pi} \cdot e_i$$

where $e_i = \prod_{k=t_i}^{T_i-1} \frac{\pi'(A_k|S_k)}{\pi'(A_k|S_k)}$

consider: on-policy control vs off-policy control w/ ϵ-greedy exploration.

what are V_* and π_* vs. $\tilde{V}_*$ and $\tilde{\pi}_*$?

V_* and π_* : off-policy

$\tilde{V}_*$ and $\tilde{\pi}_*$: on-policy