CS234 - Reinforcement Learning

A Markov Reward Process (**MRP**) is a Markov chain + rewards (everything only depends on the state $R(s_t = s)$). The **total reward** (the **return**) is

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{H-1} r_{t+H-1} = \sum_{k=0}^{H-1} \gamma^k r_{t+k}.$$

Expected return:

$$V(s) = \mathbb{E}[G_t | s_t = s] = R(s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s) V(s').$$

Let a N-dim statespace, $V = (V(s_1), \ldots, V(s_N)) \in \mathbb{R}^N$, $R, V \in \mathbb{R}^N$, and $P \in \mathbb{R}^{N \times N}$ with $P_{ij} = P(s_j|s_i)$. Then, we can write the matrix equation $V = R + \gamma PV$ and invert it $V = (I - \gamma P)^{-1} R$. Complexity of doing matrix inverse is $O(N^3)$. Otherwise, do, iteratively over k, for all $s \in S$,

$$V_k(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s) V_{k-1}(s'),\ (O(N^2) \text{ complexity})$$

A Markov Decision Process (**MDP**) is a 5-tuple (S, A, P, R, γ) where P is a transition model $P(s_{t+1} = s' | s_t = s, a_t = a)$ and R is reward function $R(s, a)$ and $\gamma \in [0, 1]$ is discount factor.

State-Value Function (expected return under π from s):

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s].$$

Optimal policy: $\pi^* = \arg\max_\pi V^\pi(s)$.

State-Action Value Function (expected return starting from s, taking action a, then following π):

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \\ &= \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a] \\ &= R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \end{aligned}$$

Bellman Backup:

$$(BV)(s) = \max_a (R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s'))$$

$$(B^\pi V)(s) = \sum_a \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right)$$

Policy Evaluation(π):

$V_0^\pi(s) \leftarrow 0$ for all s
while $\|V_{k+1}^\pi - V_k^\pi\| > \epsilon$ **do**
  forall $s \in S$ **do**
    $V_{k+1}^\pi(s) = (B^\pi V_k^\pi)(s)$.

Policy Improvement(V^π):

$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s')\ \forall s \in S, a \in A$
$\pi'(s) = \arg\max_a Q^\pi(s,a) \quad \forall s \in S$

Policy Iteration:

$\pi_0(s) \leftarrow$ whatever(A) for all s
while $\|\pi_{k+1} - \pi_k\| > \epsilon$ **do**
  $V_k^\pi \leftarrow$ PolicyEvaluation(π_k)
  $\pi_{k+1}(s) \leftarrow$ PolicyImprovement(V_k^π)

Value Iteration:

$V_0(s) \leftarrow 0$ for all s
while $\|V_{k+1} - V_k\| > \epsilon$ **do**
  forall $s \in S$ **do**
    $V_{k+1}(s) = (BV_k)(s)$

These algorithms assume Markov and require knowledge of the dynamics P. Instead, we can do (model-free) **Monte-Carlo** (requires fully observed finite episode). It is unbiased (assumes infinite-horizon problem (so that time-invariant policy) that is always eventually stopped (at the stopping time T)).

MonteCarlo(π)

$N(s) = 0$, $G(s) = 0$, $V^\pi(s) = 0$ for all s
for $i = 0, \ldots$ **do**
  sample episode $(s_{i,1}, a_{i,2}, \ldots, s_{i,T_i})$ with $a_{i,t} \sim \pi(s_t)$
  $G_{i,t} \leftarrow r_{i,t} + \gamma r_{i,t+1} + \cdots + \gamma^{T_i - 1} r_{r_i, T_i}$ for all t
  for $t = 1, \ldots, T_i$ **do**
    $s \leftarrow s_{i,t}$
    if *s visited for the **first** time in episode i* **then**
      $N(s) = N(s) + 1$
      $G(s) = G(s) + G_{i,t}$
      $V^\pi(s) = G(s)/N(s)$
      $(V^\pi(s) \leftarrow V^\pi(s) + \alpha(G_{i,t} - V^\pi(s))$ (Incremental))
      $(\Delta w = \alpha(G_{i,t} - \phi(s_{i,t})^\top w)\phi(s_{i,t})$ (MC linear))

The above is **first-visit** MC. Without the *if*, it is **every-visit** MC (which is biased but consistent and has better MSE).
Modification to get Q^π: use $N(s,a) = N(s,a) + 1$ and set $Q^\pi(s,a) = Q^\pi(s,a) + \alpha(G_{i,t} - Q^\pi(s,a))$ with $\alpha = \frac{1}{N(s,a)}$.

TD(0)(π) (instead of waiting for the full episode to finish)

$V^\pi(s) = 0$ for all s
for $i = 0, \ldots$ **do**
  sample tuple (s_t, a_t, r_t, s_{t+1}) with $a_t \sim \pi(s_t)$
  $V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\Big(\underbrace{(r_t + \gamma V^\pi(s_{t+1}))}_{\text{TD target } (\textit{bootstrap})} - V^\pi(s) \Big)$
  $(\Delta w = \alpha(r_t + \gamma\phi(s_{t+1})^\top w - \phi(s_t)^\top w)\phi(s_t)$ (TD(0) linear))

Certainty-Equivalence (**CE**) (MLE estimate for P and R)

for $t = 0, \ldots$ **do**
  sample tuple (s_t, a_t, r_t, s_{t+1}) with $a_t \sim \pi(s_t)$
  $\hat{P}(s'|s,a) = \frac{1}{N(s,a)} \sum_{r=1}^{t} \mathbb{1}(s_r = s, a_r = a, s_{r+1} = s')$
  $\hat{r}(s,a) = \frac{1}{N(s,a)} \sum_{\ell=1}^{t} \mathbb{1}(s_\ell = s, a_\ell = a) r_\ell$

SARSA and **Q-Learning** with ϵ-greedy policy

$a_0 \sim \pi(s_0)$ w.p. $1 - \epsilon$ else $a_0 \leftarrow$ random(A)
Observe (r_0, s_1)
for $t = 0, \ldots$ **do**
  Take $a_{t+1} \sim \pi(s_{t+1})$, observe (r_{t+1}, s_{t+2})
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
  $\pi(s) \leftarrow \{\arg\max_a Q(s,a)$ w.p. $1 - \epsilon$ else random$(A)\}$

SARSA (an *on-policy* algorithm) uses every element of the tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. **Q-Learning** (*off-policy*) allows learning optimum from arbitrary policies (only used for exploration) but suffers from the maximization bias. Thus, **Double Q-Learning** separates max action estimation with max value estimation: with probability 1/2, alternate between Q_1 and Q_2: for 1, set $a_{t+1}^{*1} = \arg\max_a Q_1(s_{t+1}, a)$ and evaluate
$Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, a_{t+1}^{*1}) - Q_1(s_t, a_t))$
$\pi(s) \leftarrow \{\arg\max_a (Q_1 + Q_2)(s,a)$ w.p. $1 - \epsilon$ else rand$(A)\}$

Value function approximation

S and A are assumed finite-dim. Define the loss $J(w) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}^\pi(s;w))^2]$.
Then, supervised learning: $\Delta w \overset{\text{SGD}}{=} -\frac{1}{2}\alpha \left(2\mathbb{E}_\pi[V^\pi(s) - \hat{V}^\pi(s;w)] \nabla_w \hat{V}^\pi(s;w) \right)$

Linear: $\hat{V}(s;w) = \phi(s)^\top w \implies \nabla_w \hat{V}(s;w) = \phi(s)$.

MC π evaluation: dataset $\{(s_t, G_t)\}_t$, use $V^\pi(s_t) \approx G_t$
Converges to min $\text{MSVE}_\mu(w) = \sum_s \mu(s)(V^\pi(s) - \hat{V}^\pi(s;w))^2$

TD π evaluation: dataset $\{(s_t, r_t + \gamma \hat{V}^\pi(s_{t+1}; w))\}_t$
Let $d(s)$ stationary distribution s.t. $d(s') = \sum_s \sum_a \pi(a|s) p(s'|s,a) d(s)$
Converges to min $\text{MSVE}_d(w_{TD}) \leq \frac{1}{1-\gamma} \min_w \sum_s d(s)(V^\pi(s) - \hat{V}^\pi(s;w))^2$

SARSA, **Q-Learning**, and MC control: do the same for Q:

$$\Delta w = \alpha(r + \gamma \hat{Q}(s', a'; w) - \hat{Q}(s, a; w)) \nabla_w \hat{Q}(s, a; w)$$

$$\Delta w = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; w) - \hat{Q}(s, a; w)) \nabla_w \hat{Q}(s, a; w)$$

$$\Delta w = \alpha(G_t - \hat{Q}(s_t, a_t; w)) \nabla_w \hat{Q}(s_t, a_t; w)$$

Instability: (1) function approx (2) bootstrapping (3) off-policy

Deep Q-Learning: (1) correlated samples (2) non-stationary targets are addressed via (a) experience replay (b) fixed Q-targets

for $t = 0, \ldots$ **do**
  Sample a_t given ϵ-greedy policy from $\hat{Q}(s_t, a; w)$
  Observe (r_t, s_{t+1})
  Store (s_t, a_t, r_t, s_{t+1}) in replay buffer D
  Sample minibatch from D
  for *tuple in minibatch* **do**
    $\Delta w = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; w^-) - \hat{Q}(s, a; w)) \nabla \hat{Q}(s, a; w)$
  if *sometimes* **then**
    $w^- \leftarrow w$

Double DQN even better
$\Delta w = \alpha(r + \gamma \hat{Q}(s', \arg\max_{a'} \hat{Q}(s'a'; w); w^-) - \hat{Q}(sa; w)) \nabla \hat{Q}(sa; w)$

Prioritized Experience Replay: select ith tuple such that

$$p_i = |r + \gamma \max_{a'} Q(s', a'; w^-) - Q(s, a; w)| \quad \forall i\text{th tuple in } D$$

then choose ith tuple $(s, a, r, s') \in D$ for Δw update with $\max_i p_i$.
Else, select with stochastic prioritization, where $P(i) = p_i^\beta / \left(\sum_j p_j^\beta \right)$.

Advantage Function (Dueling DQN) Features to represent V well may be different than those to compare $Q(s, a_1)$ vs $Q(s, a_2)$.

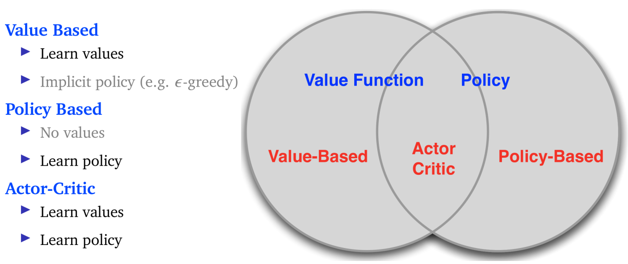
Define $\quad A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. Then, to get $\hat{Q}$:

$$\hat{Q}(s, a; w) = \hat{V}(s; w) + \left(\hat{A}(s, a; w) - \max_{a'} \hat{A}(s, a'; w) \right)$$

$$\hat{Q}(s, a; w) = \hat{V}(s; w) + \left(\hat{A}(s, a; w) - \frac{1}{A} \sum_{a'} \hat{A}(s, a'; w) \right)$$

Summary:

- **Model-based RL**
  - \+ *easy* to learn a model (supervised learning)
  - \+ learns *all there is to know* from the data
  - \- uses compute & capacity on irrelevant details
  - \- computing π (planning) non-trivial / expensive
- **Value-based RL**
  - \+ easy to get policy (e.g., $\pi(a|s) = \mathbb{1}(a = \arg\max Q(s,a))$)
  - \+ close to true objective
  - \+ fairly well-understood, good algorithms exist
  - \- still not the true objective (may focus capacity on irrelevant details, small value error can lead to larger policy error)
- **Policy-based RL**
  - \+ true objective
  - \+ easy to extend to high-dim or continuous A
  - \+ can learn stochastic policies
  - \+ policies can be simple compared to values / models
  - \- local minima
  - \- does not always generalise well



<u>Policy learning</u>: directly learn $\pi_\theta(s,a)$ minimizing $V(\theta)$.

Softmax policy: $\pi_\theta(s,a) = e^{\phi(s,a)^\top\theta} / \left(\sum e^{\phi(s,a)^\top\theta}\right)$.
Score function: $\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\pi}[\phi(s,a)]$.
Gaussian policy: $\pi_\theta(a(s) \sim \mathcal{N}(\mu_\theta(s), \sigma^2)$.
Score function: $\nabla_\theta \log \pi_\theta(s,a) = \frac{(a-\mu_\theta(s))}{\sigma^2}\nabla\mu_\theta(s)$

TODO add derivations for $\nabla V(\theta)$ to get the score function term TODO add derivations for $\nabla V(\theta)$ to get the score function term TODO add derivations for $\nabla V(\theta)$ to get the score function term

<u>Reinforce:</u> uses $\Delta\theta_t = \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$

for *each episode* $(s_1, a_1, r_1, \ldots, s_{T-1}, \ldots, r_T) \sim \pi_\theta$ **do**
  for $t = 1, \ldots, T-1$ **do**
    $\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(a_t|s_t) G_t$

Baseline $b(s)$ to reduce variance:

$$\theta_{t+1} = \theta_t + \alpha(r_{t+1} - b(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Actor-Critic: use $\hat{V}^\pi(s;w)$ (estimated) as baseline

<u>1-step Actor-Critic:</u>

for $t = 1, \ldots, T-1$ **do**
  Sample $a_t \sim \pi_\theta(a|s_t)$, observe (r_t, s_{t+1})
  $\delta_t = r_t + \gamma\hat{V}_w(s_{t+1}) - \hat{V}_w(s_t)$ (1step TDerror/advantage)
  $w \leftarrow w + \beta\delta_t\nabla_w\hat{V}_w(s_t)$ (TD(0))
  $\theta \leftarrow \theta + \alpha\delta_t\nabla_\theta \log \pi_\theta(a_t|s_t)$ (policy gradient)

Increasing stability (TRPO,PPO): add KL regularization to an older policy π_{old}: do policy gradient with $J(\theta) - \eta\text{KL}(\pi_{\text{old}}, \pi_\theta)$:

$$\text{KL}(\pi_{\text{old}}\|\pi_\theta) = \mathbb{E}_s\left[\int_A \pi_{\text{old}}(a|s)\log\frac{\pi_{\text{old}}(a|s)}{\pi_\theta(a|s)}\text{d}a\right]$$