

Lecture 11

ROB-GY 7863 / CSCI-GA 3033 7863: Planning, Learning, and Control for Space Robotics

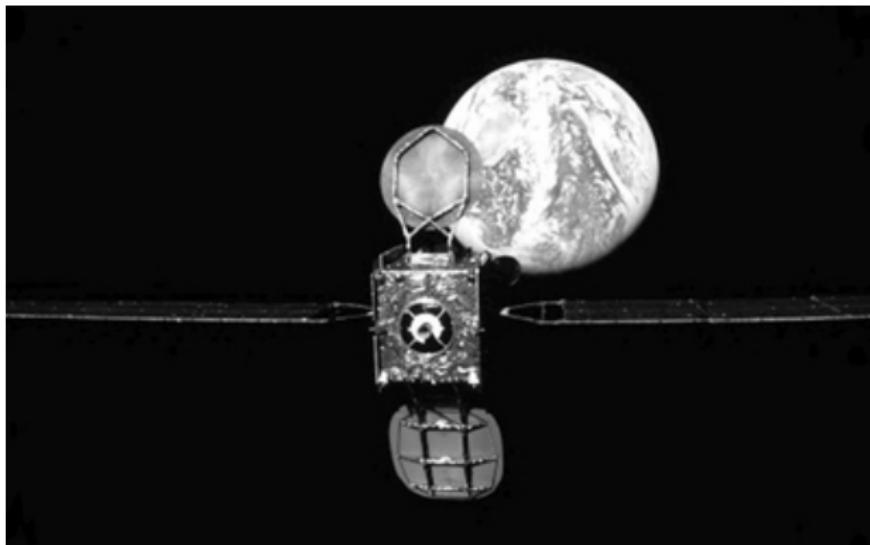
Benjamin Riviere

November 17, 2025

Logistics

- ▶ Project 2 Deadlines:
 - ▶ Final presentations: December 8th
- ▶ Grading:
 - ▶ Will post rubric this week
 - ▶ Will be similar to midterm
 - ▶ Please plan on producing a video of your simulation

Space Culture I



- ▶ Intelsat IS-901: Geostationary satellite launched in 2001 to provide internet and television over the Atlantic Ocean region

Space Culture II

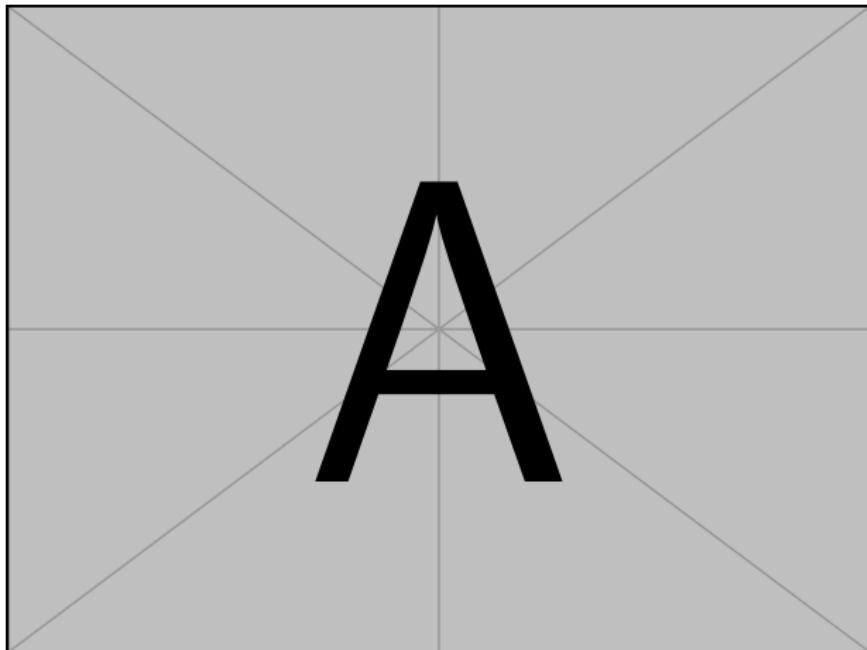


Figure: video

Space Culture III

- ▶ Mission Extension Vehicle (MEV) extends the functional lifetime of another spacecraft.
- ▶ Operation: MEV was launched by SpaceLogistics (a Northrop Grumman company) in 2019, docked with Intelsat target in 2020, repositioned in correct geostationary orbit, stationkeeping orbit until 2025, pushed target to graveyard orbit, now moving to new target, Optus D3.
- ▶ First rendezvous and docking in geostationary orbit and first docking between two commercial entities.

Space Culture IV

- ▶ MEV is a Life Extension technology because it uses its own thrusters to supply attitude control for the target
- ▶ Another approach, proposed by SIS and MDA is refueling where the servicer would open the satellite's fuel lines, refuel it, then depart.
- ▶ Life extension won the contract because it was a simpler design with less risk and more compatibility to existing satellites.
- ▶ Mission Robotic Vehicle (MRV) is the next generation mission: it replaces the docking system of MEV with a robotic arm to be able to do more missions: "detailed robotic inspection, augmentation, relocation, repair, active debris removal, and docking with non-standard client spacecraft interfaces."

Recap

- ▶ MDP: $\langle S, A, T, R, \gamma \rangle$
- ▶ Value functions
- ▶ Bellman Optimality and Evaluation Operators:

$$(\mathcal{T}V)(s) = \max_a R(s, a) + \gamma \mathbb{E}_{s'} V(s') \quad (1)$$

$$(\mathcal{T}^\pi V)(s) = \mathbb{E}_{a,s'} R(s, a) + \gamma V(s') \quad (2)$$

- ▶ Value Iteration and Policy Iteration:

$$V_{k+1} = \mathcal{T}V_k \quad (3)$$

$$\pi_{k+1} = \arg \max_a [R(s, a) + \gamma \mathbb{E}_{s'} V^{\pi_k}(s')], \quad \mathcal{T}^\pi V^\pi = V^\pi \quad (4)$$

- ▶ Overview of Reinforcement Learning
- ▶ Value-Based Reinforcement Learning:

on-policy TD learning \rightarrow SARSA \approx Policy Iteration (5)

off-policy TD learning \rightarrow Q Learning \approx Value Iteration (6)

Agenda

- ▶ Problem Extensions: POMDPs, MGs
- ▶ Policy-Based Reinforcement Learning:
 - ▶ Policy Gradient Theorem
 - ▶ REINFORCE Algorithm
 - ▶ Actor Critic Methods
 - ▶ Soft Actor Critic (SAC)

Partially Observable MDPs (POMDP) I

- ▶ Sometime we do not have access to the state and we only have access to measurements or observations from a sensor with known likelihood.
- ▶ A POMDP is an MDP with two additional components:
 - ▶ O observation space
 - ▶ Z likelihood function: $Z(s, o) = p(o|s)$
- ▶ Introduce history $h \in H$:

$$h_k = (o_1, a_1, \dots, o_k, a_k) = (o_{1:k}, a_{1:k}) \quad (7)$$

- ▶ Same policy/value setup, except history replaces the state:

$$\pi : H \rightarrow \Delta A \quad (8)$$

$$V^\pi(h) = \mathbb{E}_{s_{k+1} \sim T(\cdot | s_k, a_k), a_k \sim \pi(\cdot | h_k), o_k \sim Z(\cdot | s_k)} \left[\sum_{k=1}^K R(s_k, a_k) | h_0 = h \right] \quad (9)$$

Partially Observable MDPs (POMDP) II

- ▶ An alternative and equivalent formulation is to define the belief state (exactly the same object in the Bayes filter and Kalman filter, recall Lecture 7)

$$b_k(s_k) = p(s_k | h_k) \quad (10)$$

$$= \eta \underbrace{p(o_k | s_k)}_Z \int \underbrace{p(s_k | s_{k-1}, a_k)}_T \underbrace{p(s_{k-1} | h_{k-1})}_{b_{k-1}} ds_{k-1} \quad (11)$$

$$= \text{BayesFilter}(b_{k-1}, a_k, o_k) \quad (12)$$

Partially Observable MDPs (POMDP) III

- ▶ From any POMDP, we can construct a Belief-State MDP:

$$\text{BMDP} = \langle \mathcal{B}, A, T^b, R^b, \gamma \rangle \quad (13)$$

$$T^b(b'|b, a) = \int_{o \in O} p(o|b, a) \cdot \mathbb{I}(b' = \text{BayesFilter}(b, a, o)) \quad (14)$$

$$p(o|b, a) = \int_{s \in S} b(s) \int_{s' \in S} p(s'|s, a) p(o|s') \quad (15)$$

$$R^b(b, a) = \int_s b(s) R(s, a) \quad (16)$$

Partially Observable MDPs (POMDP) IV

- ▶ Belief-based policy and value:

$$\pi : \mathcal{B} \rightarrow \Delta U \quad (17)$$

$$V^\pi(b) = \mathbb{E}_{b_{k+1} \sim T^b(\cdot | b_k, a_k), a_k \sim \pi(\cdot | b_k)} \left[\sum_{k=1}^K \gamma^k R^b(b_k, a_k) | b_0 = b \right] \quad (18)$$

- ▶ **Example:** LQG is an POMDP with a continuous state/action space, Gaussian kernel for dynamics and likelihood and prior, and quadratic cost function. Exercise: specify the POMDP for the LQG problem.
- ▶ **Example:** RockSample is an POMDP with a discrete observation/state/action space, Exercise: specify the POMDP for the RockSample problem.

Partially Observable MDPs (POMDP) V

Figure: Rock Sample

Markov Games I

- ▶ Multi-agent MDP extension:

$$\text{MG} = \langle S, \{A^i\}_{i=1}^N, T, \{R^i\}_{i=1}^N, \gamma \rangle \quad (19)$$

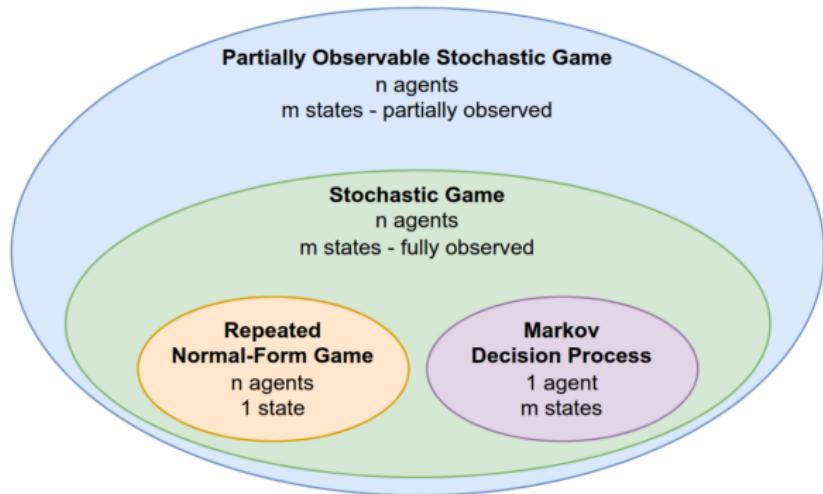
- ▶ The agents each have their own actions and rewards and share a common state and transition function.
- ▶ The set of policies $\pi = \{\pi^i\}_{i=1}^N$ is at a **Nash Equilibrium** if no agent i can improve its expected returns by changing its policy π^i , assuming the other agents policies remain fixed:

$$\forall i, \pi^i, \quad V^i(\pi^i, \pi^{-i}) \leq V^i(\pi) \quad (20)$$

where π and π^{-i} are fixed policies.

- ▶ In general, this is a challenging optimization problem and in practice, people try to avoid game theoretic settings. It is often easier to treat a team of robots as a single robot and use an MDP formulation.

Markov Games II



Markov Games III

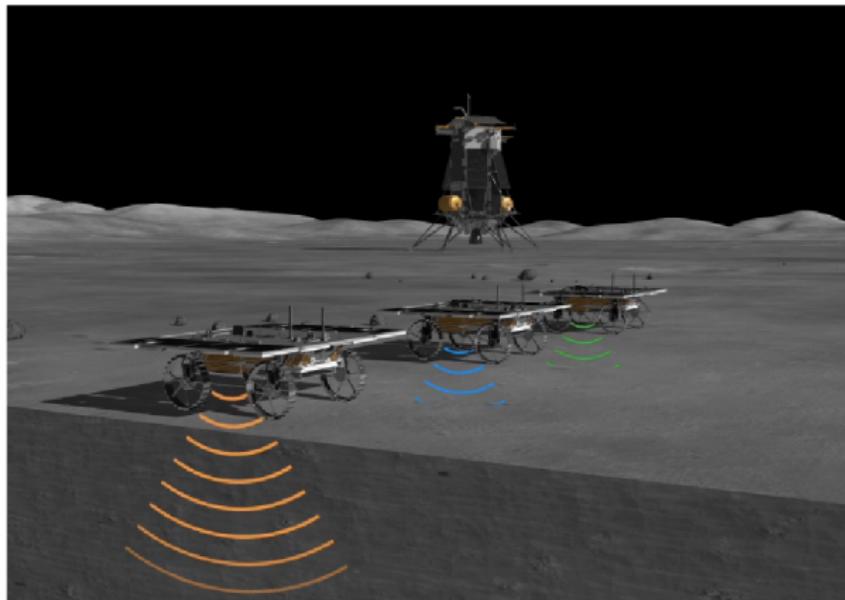


Figure: Example: CADRE mission

Motivation: Policy-Based Reinforcement Learning

Value-Based RL is great, but...:

- ▶ taking $\pi(s) = \max_a Q(s, a)$ may be difficult in continuous action spaces (although there are ways around this) **XX**.
- ▶ they may diverge if function approximation is used (deadly triad) Van Hasselt et al., [2018](#)
- ▶ they learn deterministic policies, whereas in stochastic and partially observed environments, stochastic policies are provably better Jaakkola et al., [1994](#)
- ▶ Instead, we can parameterize a policy π_θ with θ and try to directly learn θ .

Policy Gradient I

- ▶ Define a trajectory: $\tau = \{(s_k, a_k, r_k)\}_{k=1}^H$.
- ▶ Define the "trajectory reward" $G(\tau) = \sum_{k=1}^H R(s_k, a_k)$
- ▶ Define the trajectory distribution under a policy:

$$p_\theta(\tau) = \prod_{k=1}^H T(s_{k+1}|s_k, a_k) \pi_\theta(a_k|s_k) \quad (21)$$

- ▶ Define the performance of a policy:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta} G(\tau) \quad (22)$$

- ▶ How do we optimize the performance? Gradient Ascent:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta) \quad (23)$$

- ▶ How to compute $\nabla_\theta J(\theta)$?

Policy Gradient II

- ▶ Option 1: Recall these *similar* equations from Lecture 7:

$$U_{t+1} = U_t + \alpha \nabla_U J_{\lambda, \Sigma}(U_t) \approx U_t - \frac{\sum_i \exp -\frac{1}{\lambda} J(U^i) \epsilon^i}{\sum_j \exp -\frac{1}{\lambda} J(U^j)} \quad (24)$$

For example, if we pick a model class of Normal distributions with fixed covariance and learned mean, we get desired result: $\pi_\theta = \mathcal{N}(U, \Sigma)$. But, this only works for a particular model class of normal distributions.

- ▶ Option 2: "Likelihood ratio estimate" that is a general way of estimating $\nabla_\theta J(\theta)$ that can be used with neural network models.

Likelihood Ratio Estimator I

- ▶ Lets manipulate the term of interest:

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{\tau \sim p_{\theta}} G(\tau) \quad (25)$$

$$= \nabla_{\theta} \int_{\tau} p_{\theta}(\tau) G(\tau) d\tau \quad (26)$$

$$= \int_{\tau} \nabla_{\theta} p_{\theta}(\tau) G(\tau) d\tau \quad (27)$$

$$= \int_{\tau} \nabla_{\theta} \log p_{\theta}(\tau) p_{\theta}(\tau) G(\tau) d\tau \quad (28)$$

$$= \mathbb{E}_{\tau \sim p_{\theta}} \nabla_{\theta} \log p_{\theta}(\tau) G(\tau) \quad (29)$$

$$= \mathbb{E}_{\tau \sim p_{\theta}} \left(\sum_{k=1}^H \nabla_{\theta} \log \pi_{\theta}(a_k | s_k) \right) G(\tau) \quad (30)$$

Likelihood Ratio Estimator II

- ▶ This expression can be further manipulated (I am not showing this bc its just algebra):

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}} \left(\sum_{k=1}^H \gamma^{k-1} Q^{\pi}(s_k, a_k) \nabla_{\theta} \log \pi_{\theta}(a_k | s_k) \right) \quad (31)$$

- ▶ This is good because we can compute this mean with Monte Carlo Samples, and this directly leads to vanilla REINFORCE algorithm.

REINFORCE

- ▶ Algorithm:
 - ▶ Initialize parameters θ .
 - ▶ While true:
 - ▶ Sample a trajectory $\tau \sim p_\theta(\tau)$ using π_θ
 - ▶ For $k = 1, \dots, H$:
 - ▶ $G_k = \sum_{t=k}^H \gamma^{t-k} r_t$
 - ▶ $\theta \leftarrow \theta + \alpha \gamma^{k-1} G_k \nabla_\theta \log \pi_\theta(a_k | s_k)$
- ▶ Discussion: Why is this good? Easy to implement and general purpose. Why is this bad? High variance estimator

Aside: Bias and Variance in RL

- ▶ Let X be a random variable and we want to estimate its expected value $\mu = \mathbb{E}[X]$.
- ▶ A Monte Carlo Estimator is a random variable that is unbiased and has decreasing variance:

$$\hat{\mu} = \frac{1}{N} \sum_i x^i, \quad \mathbb{E}[\hat{\mu}] = \mu, \quad \text{Var}[\hat{\mu}] = \frac{\text{Var}(X)}{n} \quad (32)$$

- ▶ In the context of RL, we are usually trying to estimate $V^\pi(s)$ or $Q^\pi(s, a)$, and we have two estimators:
 - ▶ MC estimator target is $E(G_t)$, and estimator is unbiased (because $E[G_t] = V^\pi(s)$) and high variance (because $\text{Var}(G_t)$ is large).
 - ▶ TD estimator target is $E[R(s, a) + \gamma \mathbb{E}_{s'} V_k(s')]$, and estimator is biased (because $V_k(s) \neq V^\pi(s)$) and low variance (because Var is small).
- ▶ There are also **baseline** functions, which change the variance, but not the mean.

Variance Reduction Technique: Baseline I

- ▶ A baseline is any function that modifies the gradient estimate:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}} \left(\sum_{k=1}^H \gamma^{k-1} (Q^{\pi}(s_k, a_k) - b(s_k)) \nabla_{\theta} \log \pi_{\theta}(a_k | s_k) \right) \quad (33)$$

- ▶ Any function that obeys $\mathbb{E}[\nabla_{\theta} b(s)] = 0$ is a valid baseline because it does not change the mean of the estimator (this can be shown, but is not necessary for discussion).
- ▶ A common choice of baseline is $b(s) = V_{\psi}(s)$, and we can adjust the REINFORCE algorithm accordingly:
- ▶ The "advantage" function is:

$$A(s, a) = Q(s, a) - V(s) \quad (34)$$

Variance Reduction Technique: Baseline II

- ▶ REINFORCE with Value Baseline:
 - ▶ Initialize policy parameters θ and baseline parameters ψ
 - ▶ While true:
 - ▶ Sample a trajectory $\tau \sim p_\theta(\tau)$ using π_θ
 - ▶ For $k = 1, \dots, H$:
 - ▶ $G_k = \sum_{t=k}^H \gamma^{t-k} r_t$
 - ▶ $\delta_k = G_k - V_\psi(s_k)$
 - ▶ $\psi \leftarrow \psi - \alpha \delta_k \nabla_\psi V_\psi$
 - ▶ $\theta \leftarrow \theta + \alpha \gamma^{k-1} \delta_k \nabla_\theta \log \pi_\theta(a_k | s_k)$
- ▶ Use same loss for policy parameters θ . Use MSE loss for value parameters ψ :

$$L(\psi) = \sum_i (V(\psi) - G^i)^2 \quad (35)$$

$$\psi \leftarrow \psi - \alpha \nabla_\psi L = \psi - \alpha (V(\psi) - G^i) \nabla_\psi V(\psi) \quad (36)$$

- ▶ Discussion: "policy-based RL" improves by adding "value-based RL"

Actor Critic Methods

- ▶ Actor Critic methods learn from TD updates (whereas REINFORCE learns from MC rollouts)
- ▶ "Advantage" Actor Critic (A2C):
 - ▶ Initialize policy parameters θ and baseline parameters ψ
 - ▶ While true:
 - ▶ Sample action $a \sim \pi_\theta(\cdot|s)$
 - ▶ Sample next state $(s', r, done) \sim env.step(\cdot|s, a)$
 - ▶ $y_k = r + \gamma(1 - done)V_\psi(s')$
 - ▶ $\delta_k = y_k - V_\psi(s)$
 - ▶ $\psi \leftarrow \psi - \alpha \delta_k \nabla_\psi V_\psi$
 - ▶ $\theta \leftarrow \theta + \alpha \gamma^{k-1} \delta_k \nabla_\theta \log \pi_\theta(a_k|s_k)$
 - ▶ $s \leftarrow s'$
- ▶ Using TD updates also breaks "episodic" assumption, we can learn directly from online data without resetting

Generalized Advantage Estimation (GAE)

- ▶ How to interpolate between Monte Carlo updates (unbiased, high variance) and TD learning updates (biased, low variance)? Answer: Generalized Advantage Estimation.
- ▶ Define the following:

$$G_{t:t+n} = r_t + \gamma r_{t+1} + \dots + \gamma^n V_\psi(s_n) \quad (37)$$

$$A_\psi^{(n)}(s_t, a_t) = G_{t:t+n} - V_\psi(s_t) \quad (38)$$

$$A_t = \frac{\sum_{k=1}^n w^{(k)} A_\psi^{(k)}(s_t, a_t)}{\sum_{k=1}^n w^{(k)}} \quad (39)$$

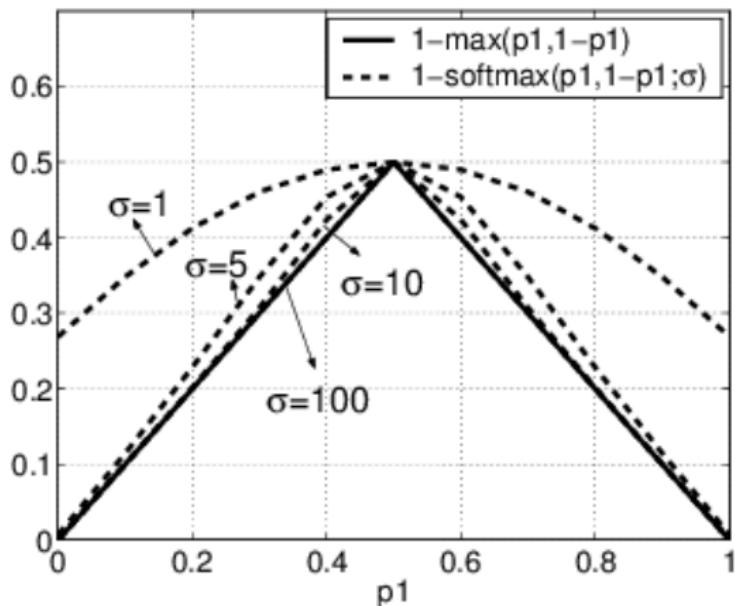
- ▶ Choose weights $w^{(k)} = \lambda^{n-1}$ to get a simple recursion:

$$\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t) \quad (40)$$

$$A_t = \delta_t + \lambda \gamma A_{t+1} \quad (41)$$

- ▶ Use that as replacement in algorithm (λ is proportional to variance and inversely proportional to bias)

Soft Actor Critic I



- ▶ What does Soft mean? Why does it help?

Soft Actor Critic II

- ▶ Maximum Entropy Reinforcement Learning:

$$J^{\text{soft}}(\theta) = \mathbb{E}_{\tau}[G(\tau) + \alpha \sum_k \mathbb{H}(\pi(.|s_k))] \quad (42)$$

where $\mathbb{H}(\pi(a_k|s_k)) = -\sum_a \pi(a|s_k) \log \pi(a|s_k)$ is the **entropy** of the policy distribution.

- ▶ Policy Evaluation:

$$\mathcal{T}^\pi Q(s, a) = R(s, a) + \gamma \mathbb{E}_{s'} V^\pi(s') \quad (43)$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi} Q(s, a) - \alpha \log \pi(a|s) \quad (44)$$

Then define V^* as the solution of a constrained optimization problem:

$$V^*(s) = \max_{\pi} \sum_a \pi(a|s) [Q^*(s, a) - \alpha \log \pi(a|s)] \text{ s.t. } \sum_a \pi(a|s) = 1 \quad (45)$$

Soft Actor Critic III

Use Lagrange Multiplies and take derivative and set to zero:

$$L(\pi, \lambda) = \sum_a \pi(a|s)[Q^*(s, a) - \alpha \log \pi(a|s)] + \lambda(1 - \sum_a \pi(a|s)) \quad (46)$$

$$\frac{\partial L}{\partial \pi} = Q^*(s, a) - \alpha(1 + \log \pi(a|s)) - \lambda = 0 \quad (47)$$

$$\pi^*(a|s) = \frac{\exp(Q^*(s, a)/\alpha)}{\sum_{a'} \exp(Q^*(s, a')/\alpha)} \quad (48)$$

implying π^* is a softmax over Q -values.

- ▶ Policy Improvement:

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(.|s_t) \parallel \frac{\exp \frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, .))}{Z^{\pi_{\text{old}}}(s_t)} \right) \quad (49)$$

Soft Actor Critic IV

- ▶ Soft Policy Improvement Theorem: $Q^{\pi_{\text{new}}}(s, a) \geq Q^{\pi_{\text{old}}}(s, a)$,
 $\forall s, a$.
- ▶ We did tabular case for concepts. General parametric case is
in Haarnoja et al., 2018.

SAC algorithm

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors $\psi, \bar{\psi}, \theta, \phi$.

for each iteration **do**

for each environment step **do**

$$\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)$$

$$\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$$

end for

for each gradient step **do**

$$\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$$

$$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i) \text{ for } i \in \{1, 2\}$$

$$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$$

$$\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$$

end for

end for

-  Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: International conference on machine learning. Pmlr, pp. 1861–1870 (cit. on p. 31).
-  Jaakkola, Tommi, Satinder Singh, and Michael Jordan (1994). "Reinforcement learning algorithm for partially observable Markov decision problems". In: Advances in neural information processing systems 7 (cit. on p. 17).
-  Van Hasselt, Hado, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil (2018). "Deep reinforcement learning and the deadly triad". In: arXiv preprint arXiv:1812.02648 (cit. on p. 17).