Soft Actor-Critic & Reinforcement Learning and Control as Probabilistic Inference

Presented by Zhanyu Wang

Department of Statistics
Purdue University

January 23, 2021

PURDUE UNIVERSITY

Soft Actor-Critic in Real World Experiments

Soft actor-critic solves these tasks quickly:

- Minitaur locomotion: 2 hours
- Block-stacking: 2 hours
- Valve-turning task from image observations: 20 hours
- Valve-turning task with the actual valve position: 3 hours
  - Prior work used PPO to learn the same task in 7.4 hours

Why formulate RL as Inference?

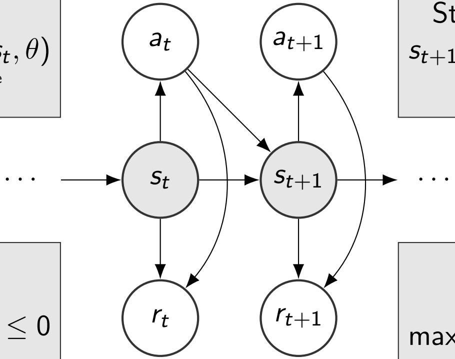
- Bayesian version of RL algorithms (changing max to softmax)
- A natural exploration strategy based on entropy maximization
- Interpretation for the reward function
- Effective tools for inverse reinforcement learning (to analyze human behavior)

Markov Decision Process (MDP)

- Probabilistic Graphical Models (PGM)
- $+$ Reward (or loss, utility) function

Policy
$a_t \sim \pi(a_t|s_t, \theta)$
Agent-define

State transitions
$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
Likely unknown



Rewards
$r_t = r(s_t, a_t) \leq 0$
Likely unknown

Goal
$\max_\pi \mathbb{E}_{s_{1:T}, a_{1:T}} \sum_{t=1}^{T} r_t$

Algorithms in Reinforcement Learning

Model-Based (RL as planning): Dynamic Programming (DP)

- Policy iteration; Value iteration.

Model-Free (RL as learning + planning):

- Monte Carlo Methods (MC)
- Temporal-Difference Learning (TD = DP + MC)
- Value-Based
  - On-policy: SARSA
  - Off-policy: Q-learning, Deep Q-Network (DeepMind, 2015)
- Policy-Based
  - Policy Gradient
  - Proximal Policy Optimization (PPO, OpenAI, 2017)
- Policy-Based + Value-Based
  - Actor-Critic
  - Deep Deterministic Policy Gradient (DDPG, Deepmind, 2015)
  - Twin Delayed Deep Deterministic PG (TD3, McGill, 2018)
  - **Soft Actor-Critic (SAC, Berkeley & Google, 2018)**

Outline

- Introduction
- Maximum entropy reinforcement learning (Levine, 2018)
  - Deterministic dynamics - Probabilistic inference
  - Stochastic dynamics - Variational inference
- Applications
  - Maximum Entropy Policy Gradients
  - Soft Q-Learning
  - Soft (Maximum Entropy) Actor-Critic
- Future directions

MDP as a Probabilistic Model

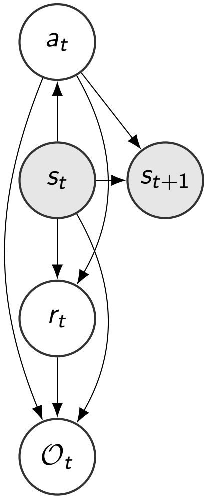
- Policy's trajectory (history) distribution

$$p(\tau) = p(s_{1:T}, a_{1:T}|\theta)$$
$$= p(s_1)\prod_{t=1}^{T} p(a_t|s_t,\theta)p(s_{t+1}|s_t,a_t)$$

- Set a binary r.v. $\mathcal{O}_t$ as optimal action indicator
  - $\mathcal{O}_t = 1$: a_t is optimal under s_t
  - $\mathcal{O}_t = 0$: not optimal
- Set the distribution of $\mathcal{O}_t$ as

$$p(\mathcal{O}_t = 1|s_t, a_t) = \exp(r(s_t, a_t))$$

- Why?

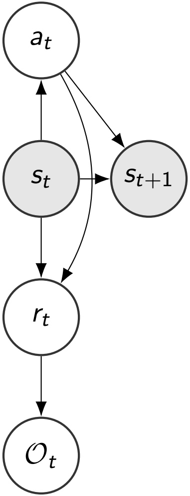


MDP as a Probabilistic Model

$$p(\tau) = p(s_1)\prod_{t=1}^{T} p(a_t|s_t,\theta)p(s_{t+1}|s_t,a_t)$$

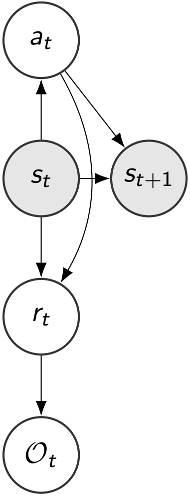
$$p(\mathcal{O}_t = 1|s_t,a_t) = \exp(r(s_t,a_t))$$

$$\begin{aligned} p(\tau|\mathcal{O}_{1:T}=1) &\propto p(\tau,\mathcal{O}_{1:T}=1) \\ &= p(s_1)\prod_{t=1}^{T} p(\mathcal{O}_t=1|s_t,a_t)p(a_t|s_t,\theta)p(s_{t+1}|s_t,a_t) \\ &= p(s_1)\prod_{t=1}^{T} \exp(r(s_t,a_t))p(a_t|s_t,\theta)p(s_{t+1}|s_t,a_t) \\ &= p(\tau)\exp\left(\sum_{t=1}^{T} r(s_t,a_t)\right) \end{aligned}$$



MDP as a Probabilistic Model

$$p(\tau|\mathcal{O}_{1:T}=1) \propto p(\tau)\exp\left(\sum_{t=1}^{T} r(s_t,a_t)\right)$$



- For deterministic dynamics ($s_{t+1}=f(s_t,a_t)$), if the initial policy is uniformly distributed ($p(a_t|s_t)=\frac{1}{|\mathcal{A}|}$), and the trajectory τ is possible, then $p(\tau)$ is constant, and we have

$$p(\tau|\mathcal{O}_{1:T}=1) \propto \exp\left(\sum_{t=1}^{T} r(s_t,a_t)\right)$$

- Trajectory with larger reward would have larger probability to be the actual history if all the actions are considered to be optimal
- Why is this useful?

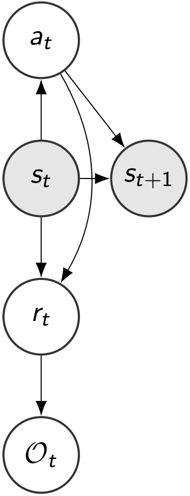
MDP as a Probabilistic Model (deterministic dynamics)

- Why is this useful?
  - Can model sub-optimal behavior (inverse RL)
  - Can apply inference algorithms to solve control and planning problems
  - Provides an explanation for why stochastic behavior might be preferred (useful for exploration and transfer learning)
- How to recover the underlying policy $\pi(a_t|s_t)$ using $\mathcal{O}_{1:T}$?

$$\pi(a_t|s_t) = p(a_t|s_t, \mathcal{O}_{t:T})$$

- Backward messages: $\beta_t(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t)$

$$\beta_t(s_t) = p(\mathcal{O}_{t:T}|s_t)$$



Policy computation using Backward messages

$$\pi(a_t|s_t) = p(a_t|s_t, \mathcal{O}_{t:T}) = \frac{p(a_t, s_t|\mathcal{O}_{t:T})}{p(s_t|\mathcal{O}_{t:T})}$$

$$= \frac{p(\mathcal{O}_{t:T}|a_t, s_t)p(s_t, a_t)}{p(\mathcal{O}_{t:T}|s_t)p(s_t)} = \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)} p(a_t|s_t)$$

$$\beta_t(s_t, a_t) = p(\mathcal{O}_t|s_t, a_t)\mathbb{E}_{s_{t+1}\sim p(s_{s+1}|s_t, a_t, \mathcal{O}_t)}[\beta_{t+1}(s_{t+1})]$$

$$\beta_t(s_t) = \mathbb{E}_{a_t\sim p(a_t|s_t)}[\beta_t(s_t, a_t)]$$

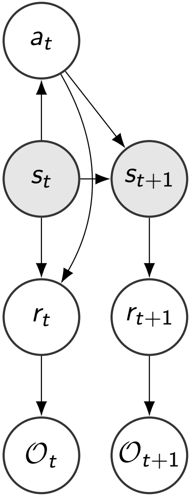
Let $V_t(s_t) = \log \beta_t(s_t)$, $Q_t(s_t, a_t) = \log \beta_t(s_t, a_t)$,

$$Q_t(s_t, a_t) = r(s_t, a_t) + \log \mathbb{E}[\exp(V_{t+1}(s_{t+1}))]$$

$$\approx r(s_t, a_t) + \max_{s_{t+1}} V_{t+1}(s_{t+1}) \quad \text{(BAD for stochastic dynamics)}$$

$$V_t(s_t) = \log \mathbb{E}[\exp(Q_t(s_t, a_t))] \approx \max_{a_t} Q_t(s_t, a_t)$$

$$\pi(a_t|s_t) = \exp(Q_t(s_t, a_t) - V_t(s_t))p(a_t|s_t)$$

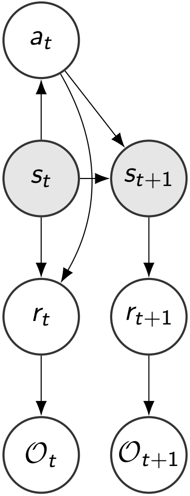


MDP as a Probabilistic Model (stochastic dynamics)

- For stochastic dynamics, we cannot control the transition probability

$$p(s_{t+1}|s_t, a_t) \neq p(s_{t+1}|s_t, a_t, \mathcal{O}_t)$$

- What do we want?
  - The trajectory distribution under $\mathcal{O}_{1:T} = 1$
- What can we change?
  - The policy parameter θ
  - The definition of $\mathcal{O}_t$
- Use $q_\theta(s_{1:T}, a_{1:T})$ to approximate $p(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T})$
- Let $x = \mathcal{O}_{1:T}, \ z = (s_{1:T}, a_{1:T})$
- Use $q_\theta(z)$ to approximate $p(z|x)$
- It's Variational Inference!



Policy computation using Variational Inference

- Evidence Lower Bound (ELBO)

$$\log p(x) \geq \mathbb{E}_{z \sim q_\theta(z)}[\log p(x, z) - \log q_\theta(z)]$$

- Let $q_\theta(s_{1:T}, a_{1:T}) = p(s_1) \prod_t p(s_{t+1}|s_t, a_t) \prod_t \pi(a_t|s_t)$

$$\log p(x, z) = \log p(s_1) + \sum_t \log p(s_{t+1}|s_t, a_t) + \sum_t (\log p(a_t|s_t) + \log p(\mathcal{O}_t|s_t, a_t))$$

- Set $r(s_t, a_t) = \log p(\mathcal{O}_t|s_t, a_t) + \log p(a_t|s_t)$ (change definition of $\mathcal{O}_t$)

$$\boxed{\log p(\mathcal{O}_{1:T}) \geq \mathbb{E}_{(s_{1:T}, a_{1:T}) \sim q} \left[\sum_t (r(s_t, a_t) - \log \pi(a_t|s_t))\right]}$$

Policy computation using Variational Inference

- Consider $t = T$

$$\mathbb{E}_{(s_T, a_T) \sim \pi(a_T|s_T) q_T(s_T)}[r(s_T, a_T) - \log \pi(a_T|s_T)]$$

- Use $\pi(a_T|s_T) \propto \exp(r(s_T, a_T))$ to maximize ELBO
- $Q(s_T, a_T) = r(s_T, a_T), \; V(s_T) = \log \int_{\mathcal{A}} \exp(Q(s_T, a_T)) da_T$

$$\pi(a_T|s_T) = \exp(Q(s_T, a_T) - V(s_T))$$

- Consider $t < T$ and maximize ELBO recursively, we have

$$\boxed{\pi(a_t|s_t) = \exp(Q(s_t, a_t) - V(s_t))}$$

$$Q(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V(s_{t+1})]$$

$$V(s_t) = \log \int_{\mathcal{A}} \exp(Q(s_t, a_t)) da_t$$

Connection between MERL and Boltzmann Exploration

$$\pi(a_t|s_t) = \frac{\exp(Q(s_t,a_t))}{\int_{\mathcal{A}} \exp(Q(s_t,a_t))da_t}$$

- A very natural exploration strategy (softmax instead of max)
- Actions with large Q-value should be taken more often
- Exploration strategy: Boltzmann-like distribution
  - -Q-function is the energy.
  - -V-function is the minimum of the expected energy w.r.t. $\pi(a_t|s_t)$ minus an entropy of $\pi(a_t|s_t)$.
- Energy-based RL with a SARSA (on-policy TD) update rule actually optimize the maximum entropy objective

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + Q(s_{t+1}, a_{t+1} \sim \pi) - Q(s_t, a_t)]$$

- Sallans and Hinton (2004)

Soft Q-Learning

- Standard Q-learning

$$\phi \leftarrow \phi + \alpha \nabla_\phi Q_\phi(s,a)(r(s,a) + V(s') - Q_\phi(s,a))$$

- Target value: $V(s) = \max_a Q_\phi(s,a)$
- Soft Q-learning

$$\phi \leftarrow \phi + \alpha \nabla_\phi Q_\phi(s,a)(r(s,a) + V(s') - Q_\phi(s,a))$$

- Target value: $V(s) = \log \int \exp(Q_{\phi'}(s,a)) \mathrm{d}a$
- Optimal policy: $\pi(a|s) = \exp(Q_\phi(s,a) - V(s))$
- General equivalence between soft Q-learning and policy gradients (Haarnoja et al., 2017)

Maximum Entropy Policy Gradients

- Similar to REINFORCE (Williams, 1992) which maximizes the value function $V_{\pi_\theta}(s_0)$
- Directly maximize the ELBO

$$J(\theta) = \mathbb{E}_{(s_{1:T}, a_{1:T}) \sim q} \left[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t|s_t)) \right]$$

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim q_t(s_t, a_t)} \left[\nabla_\theta \log q_\theta(a_t|s_t) \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - \log q_\theta(a_{t'}|s_{t'}) - 1 \right) \right]$$

$$= \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim q_t(s_t, a_t)} \left[\nabla_\theta \log q_\theta(a_t|s_t) \hat{A}(s_t, a_t) \right]$$

- Use generalized advantage estimator (Schulman et al., 2016)

Soft Actor-Critic Algorithms (Haarnoja et al., 2018)

- Standard Actor-Critic (AC) models both policy ($\pi_\phi(a_t|s_t)$) and Q-function ($Q_\theta(s_t, a_t)$)
- Maximum Entropy (soft) Actor-Critic is based on soft Policy Improvement and soft Q-function
- Let $V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$
- Minimize $J_Q(\theta)$ (critic), $J_\pi(\phi)$ (actor)

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\bar{\theta}}(s_{t+1})]))^2\right]$$

$$\begin{aligned}\nabla_\theta J_Q(\theta) = {} & \nabla_\theta Q_\theta(a_t, s_t)(Q_\theta(a_t, s_t) - \\ & (r(s_t, a_t) + \gamma Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log(\pi_\phi(a_{t+1}|s_{t+1}))))\end{aligned}$$

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}[\mathbb{E}_{a_t\sim\pi_\phi}[\alpha \log(\pi_\phi(a_t, s_t)) - Q_\theta(s_t, a_t)]], a_t = f_\phi(\epsilon_t; s_t)$$

$$\begin{aligned}\nabla_\phi J_\pi(\phi) = {} & \nabla_\phi \alpha \log(\pi_\phi(a_t|s_t)) + \\ & (\nabla_{a_t} \alpha \log(\pi_\phi(a_t|s_t)) - \nabla_{a_t} Q(s_t, a_t))\nabla_\phi f_\phi(\epsilon_t; s_t)\end{aligned}$$

Soft Actor-Critic Algorithms

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters
$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
  end for
end for
Output: θ_1, θ_2, ϕ ▷ Optimized parameters

- Use two soft Q-functions to mitigate positive bias in the policy improvement step that is known to degrade performance of value based methods.
- Also optimize temperature α.

Soft Actor-Critic Experiments

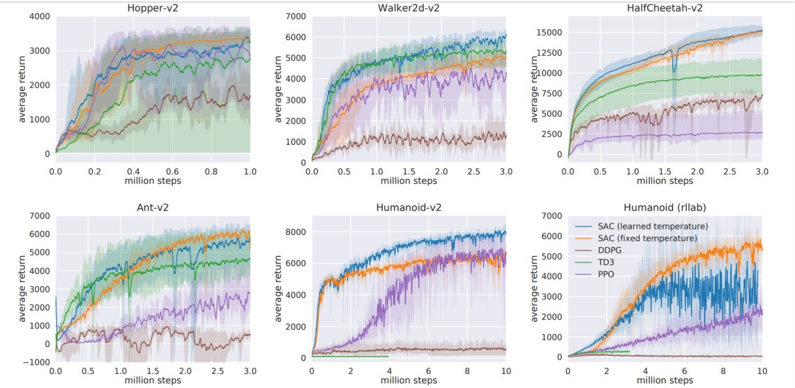


Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

Soft Actor-Critic Hyperparameters

Table 1: SAC Hyperparameters

Parameter	Value
optimizer	Adam (Kingma & Ba, 2015)
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
entropy target	$-\dim(\mathcal{A})$ (e.g. , -6 for HalfCheetah-v1)
nonlinearity	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1

Soft Actor-Critic in Real World Experiments

Soft actor-critic solves these tasks quickly:

- Minitaur locomotion: 2 hours
- Block-stacking: 2 hours
- Valve-turning task from image observations: 20 hours
- Valve-turning task with the actual valve position: 3 hours
    - Prior work used PPO to learn the same task in 7.4 hours

Benefits of soft optimality

- Improve exploration and prevent entropy collapse
  - important for policy gradient algorithms
- Easier to specialize (finetune) policies for more specific tasks
- Principled approach to break ties
  - equally good actions get equal probability
- Better robustness (due to wider coverage of states)
  - inject noise to the policy
- Can reduce to hard optimality as reward magnitude increases
- Good model for modeling human behavior (inverse RL (Ziebart et al., 2008))

Conclusion: connection between RL and Inference

- RL viewed as inference in graphical model
  - Set value function and Q-function as backward messages
  - Maximize both reward and entropy
  - Use variational inference to remove optimism
  - Use $\mathcal{O}_t$ to make probabilistic interpretation of rewards
- Applications
  - Soft Q-Learning
  - Maximum Entropy Policy Gradients
  - Maximum Entropy Actor-Critic Algorithm
- Future Directions
  - Maximum entropy RL and Latent Variable models
    - Graphical model with additional variables to model time correlated stochasticity for exploration
    - Higher-level control through learned latent action spaces
  - Are maximum Entropy methods robust to domain shift, unexpected perturbations, and model errors?
  - Re-design of reward functions

References

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Sallans, B. and Hinton, G. E. (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5(Aug):1063–1088.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. *ICLR*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA.

Further reading

- Sergey Levine. CS285 at UC Berkeley. http://rail.eecs.berkeley.edu/deeprlcourse/
- Sergey Bartunov. Reinforcement learning through the lense of variational inference. *DeepBayes2018*. https://www.youtube.com/watch?v=6v3RxQycT0E
- Softlearning (official SAC GitHub repo). https://github.com/rail-berkeley/softlearning
- Lilian Weng. Policy Gradient Algorithms. https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html