CS885 Reinforcement Learning
Module 2: June 6, 2020

Maximum Entropy Reinforcement Learning

Haarnoja, Tang et al. (2017) Reinforcement Learning with Deep Energy Based Policies, *ICML*.

Haarnoja, Zhou et al. (2018) Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, *ICML*.

Reinforcement Learning

Deterministic Policies

- There always exists an optimal deterministic policy
- Search space is smaller for deterministic than stochastic policies
- Practitioners prefer deterministic policies

Stochastic Policies

- Search space is continuous for stochastic policies (helps with gradient descent)
- More robust (less likely to overfit)
- Naturally incorporate exploration
- Facilitate transfer learning
- Mitigate local optima

Encouraging Stochasticity

Standard MDP

- States: S
- Actions: A
- Reward: $R(s,a)$
- Transition: $\Pr(s'|s,a)$
- Discount: γ

Soft MDP

- States: S
- Actions: A
- Reward: $R(s,a) + \lambda H\big(\pi(\cdot\,|s)\big)$
- Transition: $\Pr(s'|s,a)$
- Discount: γ

Optimal Policy

- Standard MDP

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{n=0}^{N} \gamma^n E_{s_n,a_n|\pi}[R(s_n, a_n)]$$

- Soft MDP

$$\pi^*_{soft} = \underset{\pi}{\operatorname{argmax}} \sum_{n=0}^{N} \gamma^n E_{s_n,a_n|\pi}\left[R(s_n, a_n) + \lambda H\big(\pi(\cdot\,|s_n)\big)\right]$$

Maximum entropy policy
Entropy regularized policy

Q-function

- Standard MDP

$$Q^{\pi}(s_0, a_0) = R(s_0, a_0) + \sum_{n=1}^{\infty} \gamma^n E_{s_n, a_n | s_0, a_0, \pi}[R(s_n, a_n)]$$

- Soft MDP

$$Q^{\pi}_{soft}(s_0, a_0) = R(s_0, a_0) + \sum_{n=1}^{\infty} \gamma^n E_{s_n, a_n | s_0, a_0, \pi}\left[R(s_n, a_n) + \lambda H\big(\pi(\cdot \mid s_n)\big)\right]$$

NB: No entropy with first reward term
since action is not chosen according to π

Greedy Policy

- Standard MDP (deterministic policy)

$$\pi_{greedy}(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

- Soft MDP (stochastic policy)

$$\pi_{greedy}(\cdot|s) = \underset{\pi}{\operatorname{argmax}} \sum_a \pi(a|s)Q(s,a) + \lambda H\big(\pi(\cdot|s)\big)$$

$$= \frac{\exp(Q(s,\cdot)/\lambda)}{\sum_a \exp(Q(s,a)/\lambda)} = softmax(Q(s,\cdot)/\lambda)$$

when $\lambda \rightarrow 0$ then $softmax$ becomes regular max

Soft Policy Iteration

SoftPolicyIteration(MDP, λ)

Initialize π_0 to any policy

$i \leftarrow 0$

Repeat

Policy evaluation:

Repeat until convergence

$$Q^{\pi_i}_{soft}(s,a) \leftarrow R(s,a)$$
$$+\gamma \sum_{s'} \Pr(s'|s,a) \left[\sum_{a'} \pi_i(a'|s') Q^{\pi_i}_{soft}(s',a') + \lambda H\big(\pi_i(\cdot\,|s')\big)\right] \forall s,a$$

Policy improvement:

$$\pi_{i+1}(a|s) \leftarrow softmax\left(Q^{\pi_i}_{soft}(s,a)/\lambda\right) = \frac{\exp\left(Q^{\pi_i}_{soft}(s,a)/\lambda\right)}{\sum_{a'} \exp\left(Q^{\pi_i}_{soft}(s,a')/\lambda\right)} \ \forall s,a$$

$i \leftarrow i+1$

Until $\left\|Q^{\pi_i}_{soft}(s,a) - Q^{\pi_{i-1}}_{soft}(s,a)\right\|_\infty \leq \epsilon$

Soft Actor-Critic

- RL version of soft policy iteration
- Use neural networks to represent policy and value function
- At each policy improvement step, project new policy in the space of parameterized neural nets

Soft Actor Critic (SAC)

Initialize weights $\boldsymbol{w}$, $\overline{\boldsymbol{w}}$, θ at random in $[-1,1]$
Observe current state s
Loop
    Sample action $a \sim \pi_\theta(\cdot\,|s)$ and execute it
    Receive immediate reward r
    Observe new state s'
    Add (s, a, s', r) to experience buffer
    Sample mini-batch of experiences from buffer
    For each experience $(\hat{s}, \hat{a}, \hat{s}', \hat{r})$ in mini-batch
        Sample $\hat{a}' \sim \pi_\theta(\cdot\,|\hat{s}')$
        Gradient: $\frac{\partial Err}{\partial \boldsymbol{w}} = \left[Q_{\boldsymbol{w}}^{soft}(\hat{s}, \hat{a}) - \hat{r} - \gamma [Q_{\overline{\mathbf{w}}}^{soft}(\hat{s}', \hat{a}') + \lambda H(\pi_\theta(\cdot\,|\hat{s}'))]\right] \frac{\partial Q_{\boldsymbol{w}}^{soft}(\hat{s}, \hat{a})}{\partial \boldsymbol{w}}$
        Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\partial Err}{\partial \boldsymbol{w}}$
        Update policy: $\theta \leftarrow \theta - \alpha \frac{\partial KL\left(\pi_\theta \middle| softmax\left(Q_{\overline{w}}^{soft}/\lambda\right)\right)}{\partial \theta}$
    Update state: $s \leftarrow s'$
    Every c steps, update target: $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{w}$