Energy-Based (Entropy-Based) RL Models

Hong Xingxing

December 13, 2018

Outline

1. Maximum Entropy Inverse Reinforcement Learning
2. Reinforcement Learning with Deep Energy-Based Policies

Motivations

Both IRL and the matching of feature counts are ambiguous.

- Each policy can be optimal for many reward functions.
- Many policies lead to the same feature counts.

This paper employ the principle of *maximum entropy* to resolve the ambiguity in choosing a distribution over decisions.

Notations

- S is a finite set of N states.
- $A = \{a_1, .., a_k\}$ is a set of k actions.
- $P_{sa}(s')$ is the state transition probability of landing at state s': $P(s, a, s')$ upon taking the action a at state s.
- $\gamma \in [0, 1)$ is the discount factor.
- $R : S \rightarrow \mathbf{R}$ is the reward function.

Max Entropy

- $\zeta = \{(s, a)\}$ is a trajectory.
- $\mathbf{f_s} \in \mathbf{R}^k$ is the feature vector of the state s.
- $\theta \in \mathbf{R}^k$ reward function parameters.
- $P(\zeta)$ probability of the trajectory ζ to occur
- $P(s)$ the probability of visiting state s (state visitation frequency), $P(\zeta) = \prod_{s \in \zeta} P(s)$.

Assumptions

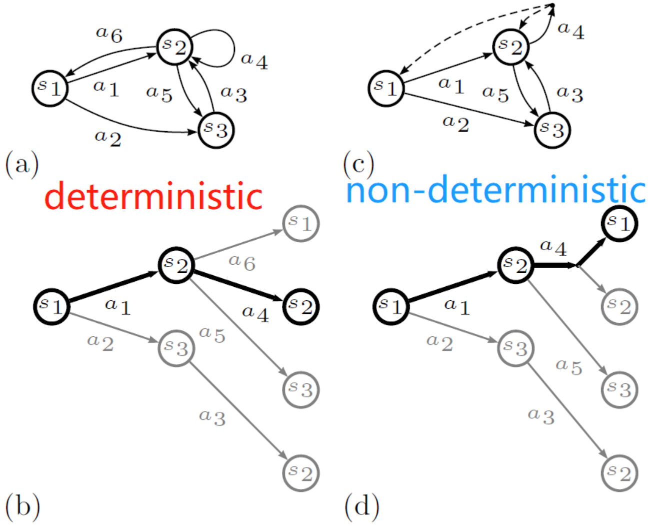
- The reward value of a trajectory is simply the sum of state rewards, or, equivalently, the reward weight applied to the path feature counts, $\mathbf{f}_\zeta = \sum_{s_j \in \zeta} \mathbf{f}_{s_j}$, which are the sum of the state features along the path.

$$\text{reward}\left(\mathbf{f}_\zeta\right) = \theta^\top \mathbf{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \mathbf{f}_{s_j} \tag{1}$$

- Apply the principle of maximum entropy, choosing the distribution that does not exhibit any additional preferences beyond matching feature expectations. Plans with equivalent rewards have equal probabilities, and plans with higher rewards are exponentially more preferred.

$$P(\zeta) \propto e^{\text{reward}\left(\mathbf{f}_\zeta\right)} \tag{2}$$

deterministic and non-deterministic MDP



Deterministic Path Distributions

The resulting distribution over paths for deterministic MDPs is parameterized by reward weights θ

$$P\left(\zeta_i|\theta\right) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^\top \mathbf{f}_{s_j}} \tag{3}$$

$Z(\theta)$ is a partition function

Non-Deterministic Path Distributions

In general MDPs, actions produce non-deterministic transitions between states (Figure 1c) according to the state transition distribution, T.

- The space of action outcomes $\mathcal{T}$
- An outcome sample, o, specifying the next state for every action
- The indicator function, $I_{\zeta \in o}$ is 1 when ζ is compatible with o and 0 otherwise

$$\begin{aligned} P(\zeta|\theta, T) &= \sum_{o \in \mathcal{T}} P_T(o) \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, o)} I_{\zeta \in o} \\ &\approx \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, T)} \prod_{s_{t+1}, a_t, s_{t \in \zeta}} P_T\left(s_{t+1} | a_t, s_t\right) \end{aligned}$$

Learning from Demonstrated Behavior

$$\begin{aligned}
\theta^* = \text{argmax}_\theta L(\theta) &= \text{argmax}_\theta \frac{1}{M} \log P(\{\zeta\}|\theta) \\
&= \text{argmax}_\theta \frac{1}{M} \log \prod_\zeta P(\zeta|\theta) \\
&= \text{argmax}_\theta \frac{1}{M} \sum_\zeta \log P(\zeta|\theta) \\
&= \text{argmax}_\theta \frac{1}{M} \sum_\zeta \log \frac{e^{R(\zeta)}}{Z} \\
&= \text{argmax}_\theta \frac{1}{M} \sum_\zeta \Big(R(\zeta) - \log Z\Big)
\end{aligned}$$

Where M is the number of trajectories, Z is the normalization term $Z = \sum_\zeta e^{R(\zeta)}$

Learning from Demonstrated Behavior

$$\theta^* = \text{argmax}_\theta \frac{1}{M} \sum_{\zeta} \left(R(\zeta) - \log \sum_{\zeta} e^{R(\zeta)} \right)$$

$$\theta^* = \text{argmax}_\theta \frac{1}{M} \sum_{\zeta} \left(\theta^T \mathbf{f}_\zeta - \log \sum_{\zeta} e^{\theta^T \mathbf{f}_\zeta} \right)$$

Learning from Demonstrated Behavior

$$\begin{aligned}
\nabla_\theta L &= \frac{1}{M}\sum_{\zeta}\left(\mathbf{f}_\zeta - \frac{1}{\sum_\zeta e^{R(\zeta)}}\sum_{\zeta}(e^{R(\zeta)}\tfrac{dR(\zeta)}{d\theta})\right)\\
&= \frac{1}{M}\sum_{\zeta}\left(\mathbf{f}_\zeta - \frac{1}{\sum_\zeta e^{R(\zeta)}}\sum_{\zeta}(e^{R(\zeta)}\frac{dR(\zeta)}{d\theta})\right)\\
&= \frac{1}{M}\sum_{\zeta}\left(\mathbf{f}_\zeta - \sum_{\zeta}\frac{e^{R(\zeta)}}{\sum_\zeta e^{R(\zeta)}}\mathbf{f}_\zeta\right)\\
&= \frac{1}{M}\sum_{\zeta}\left(\mathbf{f}_\zeta - \sum_{\zeta}P(\zeta|\theta)\mathbf{f}_\zeta\right)\\
&= \tilde{\mathbf{f}} - \sum_{\zeta}P(\zeta|\theta,T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i}D_{s_i}\mathbf{f}_{s_i}
\end{aligned}$$

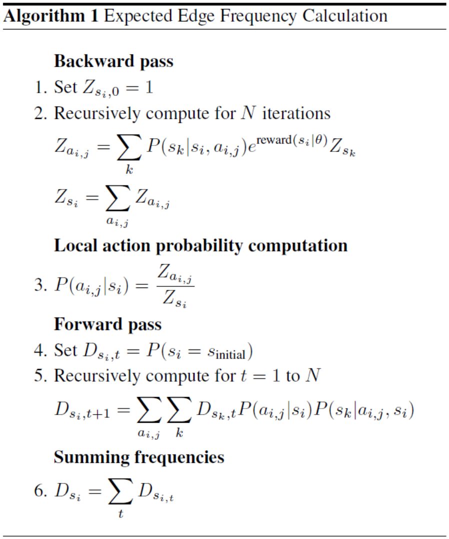
D_{si} is expected state visitation frequencies

Application: Driver Route Modeling

Motivated by applications of imitating learning of driver route choices

- Road networks present a large planning space with known structure
- A deterministic MDP with over 300000 sates (road segments) and 900000 actions (transitions at intersections)
- The drivers are attempting to reach some goal while efficiently optimizing some trade-off between *time, safety, stress, fuel costs, maintenance costs*, this value is called *cost*
- Path features (road type, speed, lanes, and transitions)
- In maximum entropy model setting, the value is simply the optimal path cost to the goal after taking a particular action $P(action a|s_i, \theta) \propto e^{Q^*(s_i,a)}$

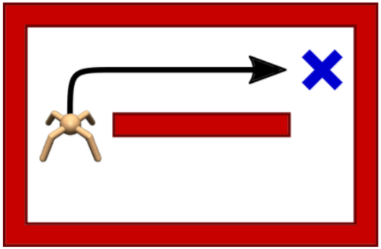
Expected State Visitation Frequencies Calculation



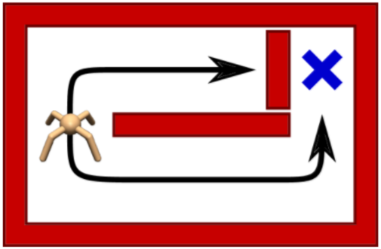
```
Algorithm 1 Expected Edge Frequency Calculation

    Backward pass
1.  Set $Z_{s_i,0} = 1$
2.  Recursively compute for $N$ iterations
    $Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$
    $Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$
    Local action probability computation
3.  $P(a_{i,j}|s_i) = \frac{Z_{a_{i,j}}}{Z_{s_i}}$
    Forward pass
4.  Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5.  Recursively compute for $t = 1$ to $N$
    $D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$
    Summing frequencies
6.  $D_{s_i} = \sum_t D_{s_i,t}$
```

Maze Game



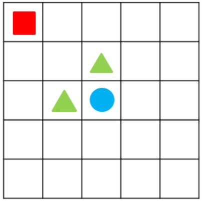
2a



2b

- Discrete or continuous state-action space?
- Adaptive model or retrain the model?

Shortest Path Problem via Q-Learning

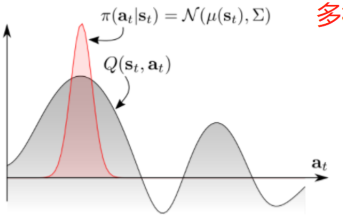


- 5×5 grid world
- red indicates starting point
- blue indicates goal point
- green indicates walls
- find the shortest path

soft Q-learning

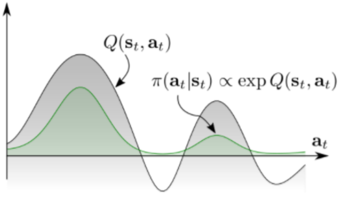
- Learning maximum entropy policies
- Expresses the optimal policy via a Boltzmann distribution
- Intuitively, framing control as inference produces policies that aim to capture not only the single deterministic behavior that has the lowest cost, but the entire range of low-cost behaviors, explicitly maximizing the entropy of the corresponding policy

Multi Modal Q function



3a

多模态Q函数



3b

$$\pi\left(\mathbf{a}_t | \mathbf{s}_t\right) \propto \exp\left(-\mathcal{E}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right) \tag{4}$$

Maximum Entropy RL Objective

- Standard reinforcement learning objective

$$\pi^*_{\mathrm{std}} = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \tag{5}$$

- Maximum entropy RL augments the reward with an entropy term

$$\pi^*_{\mathrm{MaxEnt}} = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \alpha\mathcal{H}(\pi(\cdot|\mathbf{s}_t))\right] \tag{6}$$

- **greedy - exploration?**
- **entropy regularizer - Boltzmann distribution?**
- **What if $\alpha \longrightarrow 0$?**

Theorem 1: optimal policy π^*_{MaxEnt}

soft Q-function definition

$$Q^*_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}_t\right) = r_t + \mathbb{E}_{(\mathbf{s}_{t+1},\ldots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l\left(r_{t+l} + \alpha\mathcal{H}\left(\pi^*_{\mathrm{MaxEnt}}(\cdot|\mathbf{s}_{t+l})\right)\right)\right] \tag{7}$$

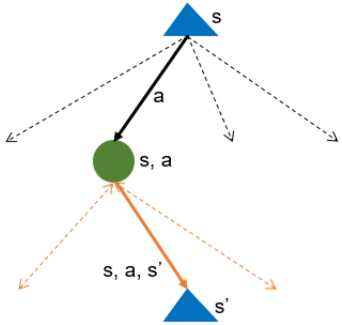
soft value function definition

$$V^*_{\mathrm{soft}}\left(\mathbf{s}_t\right) = \alpha\log\int_{\mathcal{A}}\exp\left(\frac{1}{\alpha}Q^*_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}'\right)\right)d\mathbf{a}' \tag{8}$$

optimal policy π^*_{MaxEnt} is given by

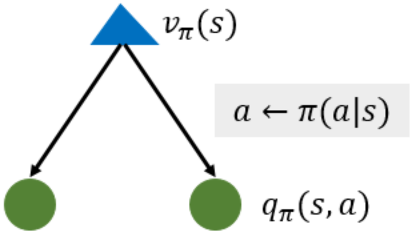
$$\pi^*_{\mathrm{MaxEnt}}\left(\mathbf{a}_t|\mathbf{s}_t\right) = \exp\left(\frac{1}{\alpha}\left(Q^*_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - V^*_{\mathrm{soft}}\left(\mathbf{s}_t\right)\right)\right) \tag{9}$$

Standard Bellman Equation-v node and q node



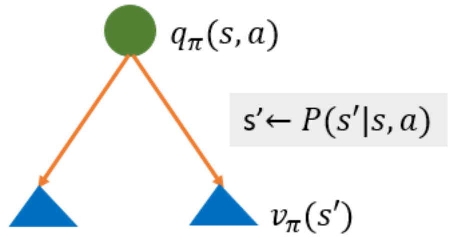
- $\pi(a|s)$
- $P(s'|s,a)$
- $R(s,a,s')$
- $v_\pi(s)$
- $q_\pi(s,a)$

Standard Bellman Equation-v node



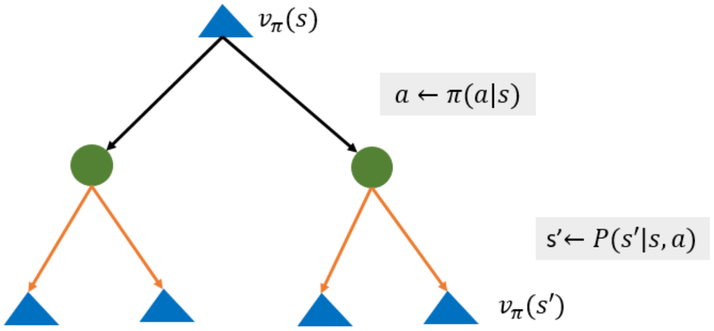
$$v_\pi(s) = \sum_{a \in A(s)} \pi(a|s)\, q_\pi(s,a)$$

Standard Bellman Equation-q node



$$q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)\ v_\pi(s')$$

Standard Bellman Equation-$v(s)$node and $v(s')$ node



$$v_\pi(s) = \sum_{a \in A(s)} \pi(a|s) \left[R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)\, v_\pi(s') \right]$$

Energy-Based models and π^*_{MaxEnt}

There is a close connection between such energy-based models and soft versions of value functions and Q-functions

$$\pi^*_{\mathrm{MaxEnt}}\left(\mathbf{a}_t|\mathbf{s}_t\right) = \exp\left(\frac{1}{\alpha}\left(Q^*_{\mathrm{soft}}\left(\mathbf{s}_t,\mathbf{a}_t\right) - V^*_{\mathrm{soft}}\left(\mathbf{s}_t\right)\right)\right)$$
$$\propto \exp\left(-\left(-\frac{1}{\alpha}Q^*_{\mathrm{soft}}\right)\right)$$

We set $\mathcal{E}\left(\mathbf{s}_t,\mathbf{a}_t\right) = -\frac{1}{\alpha}Q_{\mathrm{soft}}\left(\mathbf{s}_t,\mathbf{a}_t\right)$

Theorem 2: soft Bellman equation

$$Q^*_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}_t\right) = r_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathrm{s}}}\left[V^*_{\mathrm{soft}}\left(\mathbf{s}_{t+1}\right)\right] \tag{10}$$

Theorem 3: Soft Q-Iteration

Soft Q-iteration. Let $Q_{\mathrm{soft}}(\cdot,\cdot)$ and $V_{\mathrm{soft}}(\cdot)$ be bounded and assume that $\int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\mathrm{soft}}\left(\cdot, \mathbf{a}'\right)\right) d\mathbf{a}' < \infty$ and that $Q^*_{\mathrm{soft}} < \infty$ exists. Then the fixed-point iteration

$$Q_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}_t\right) \leftarrow r_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}}\left[V_{\mathrm{soft}}\left(\mathbf{s}_{t+1}\right)\right], \forall \mathbf{s}_t, \mathbf{a}_t$$
$$V_{\mathrm{soft}}\left(\mathbf{s}_t\right) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\mathrm{soft}}\left(\mathbf{s}_t, \mathbf{a}'\right)\right) d\mathbf{a}', \forall \mathbf{s}_t$$

converges to Q^*_{soft} and V^*_{soft} respectly

Soft Q-Learning

- Express the soft value function in terms of an expectation via **importance sampling**

$$V_{\mathrm{soft}}^{\theta}\left(\mathbf{s}_t\right) = \alpha \log \mathbb{E}_{q_{\mathrm{a}'}}\left[\frac{\exp\left(\frac{1}{\alpha} Q_{\mathrm{soft}}^{\theta}\left(\mathbf{s}_t, \mathbf{a}'\right)\right)}{q_{\mathbf{a}'}\left(\mathbf{a}'\right)}\right] \qquad (11)$$

- Express the soft Q-iteration in an equivalent form as minimizing

$$J_Q(\theta) = \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}, \mathbf{a}_t \sim q_{\mathbf{a}_t}}\left[\frac{1}{2}\left(\hat{Q}_{\mathrm{soft}}^{\bar{\theta}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - Q_{\mathrm{soft}}^{\theta}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)^2\right] \qquad (12)$$

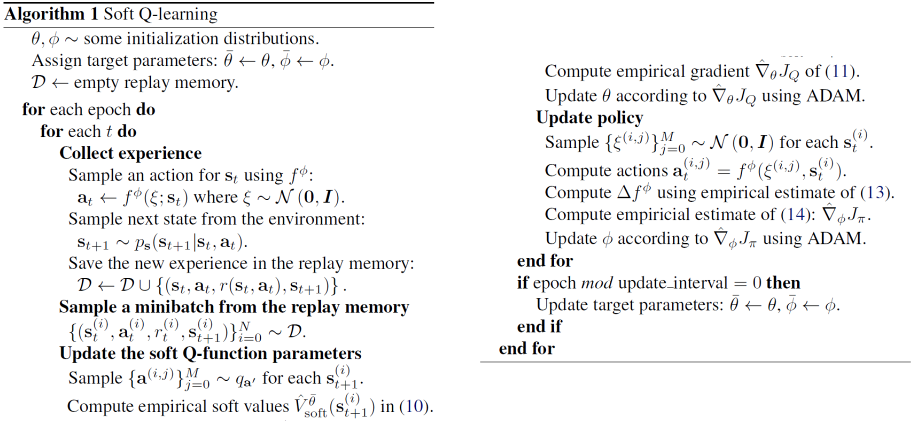
combine importance sampling and deep Q network approximator

Stein Variational Gradient Descent (SVGD): sample from $\exp\left(\frac{1}{\alpha} Q_{\mathrm{soft}}^{\theta}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)$

Denote the induced distribution of the actions as $\pi^{\phi}\left(\mathbf{a}_t|\mathbf{s}_t\right)$, and we want to find parameters ϕ so that the induced distribution

$$J_{\pi}\left(\phi; \mathbf{s}_t\right) = \mathrm{D}_{\mathrm{KL}}\left(\pi^{\phi}(\cdot|\mathbf{s}_t) \middle\| \exp\left(\frac{1}{\alpha}\left(Q_{\mathrm{soft}}^{\theta}\left(\mathbf{s}_t, \cdot\right) - V_{\mathrm{soft}}^{\theta}\right)\right)\right) \tag{13}$$

Soft Q Learning



Algorithm 1 Soft Q-learning

```
  $\theta, \phi \sim$ some initialization distributions.
  Assign target parameters: $\bar{\theta} \leftarrow \theta$, $\bar{\phi} \leftarrow \phi$.
  $\mathcal{D} \leftarrow$ empty replay memory.
  for each epoch do
    for each $t$ do
      Collect experience
        Sample an action for $\mathbf{s}_t$ using $f^\phi$:
          $\mathbf{a}_t \leftarrow f^\phi(\xi; \mathbf{s}_t)$ where $\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
        Sample next state from the environment:
          $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.
        Save the new experience in the replay memory:
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ .
      Sample a minibatch from the replay memory
        $\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^N \sim \mathcal{D}$.
      Update the soft Q-function parameters
        Sample $\{\mathbf{a}^{(i,j)}\}_{j=0}^M \sim q_{\mathbf{a}'}$ for each $\mathbf{s}_{t+1}^{(i)}$.
        Compute empirical soft values $\hat{V}_{\text{soft}}^{\bar{\theta}}(\mathbf{s}_{t+1}^{(i)})$ in (10).
        Compute empirical gradient $\hat{\nabla}_\theta J_Q$ of (11).
        Update $\theta$ according to $\hat{\nabla}_\theta J_Q$ using ADAM.
      Update policy
        Sample $\{\xi^{(i,j)}\}_{j=0}^M \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for each $\mathbf{s}_t^{(i)}$.
        Compute actions $\mathbf{a}_t^{(i,j)} = f^\phi(\xi^{(i,j)}, \mathbf{s}_t^{(i)})$.
        Compute $\Delta f^\phi$ using empirical estimate of (13).
        Compute empiricial estimate of (14): $\hat{\nabla}_\phi J_\pi$.
        Update $\phi$ according to $\hat{\nabla}_\phi J_\pi$ using ADAM.
    end for
    if epoch $mod$ update_interval $= 0$ then
      Update target parameters: $\bar{\theta} \leftarrow \theta$, $\bar{\phi} \leftarrow \phi$.
    end if
  end for
```

Examples

`https://www.jiqizhixin.com/articles/2017-10-14-7`

References

- `https://www.jiqizhixin.com/articles/2017-10-14-7`
- Maximum Entropy Inverse Reinforcement Learning
- Reinforcement Learning with Deep Energy-Based Policies
- `http://178.79.149.207/posts/maxent.html`
- `https://www.youtube.com/watch?v=7Nm1N6sUoVs`
- `https://arxiv.org/pdf/1704.06440.pdf`Equivalence Between Policy Gradients and Soft Q-Learning
- SoftQ Learning Source Code