Extreme Q-Learning: MaxEnt RL without Entropy

Divyansh Garg[1], **Joey Hejna**[1], Matthieu Geist[2], Stefano Ermon[1]
[1]Stanford University
[2]Google Brain
Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)

2025.03.06.
에이전트브레인스토밍 스터디
김동민

https://arxiv.org/abs/2301.02328

Andrew Barto and Richard Sutton Receive 2024 ACM A.M. Turing Award

- announced Mar 5, 2025!

AWARDS & RECOGNITION

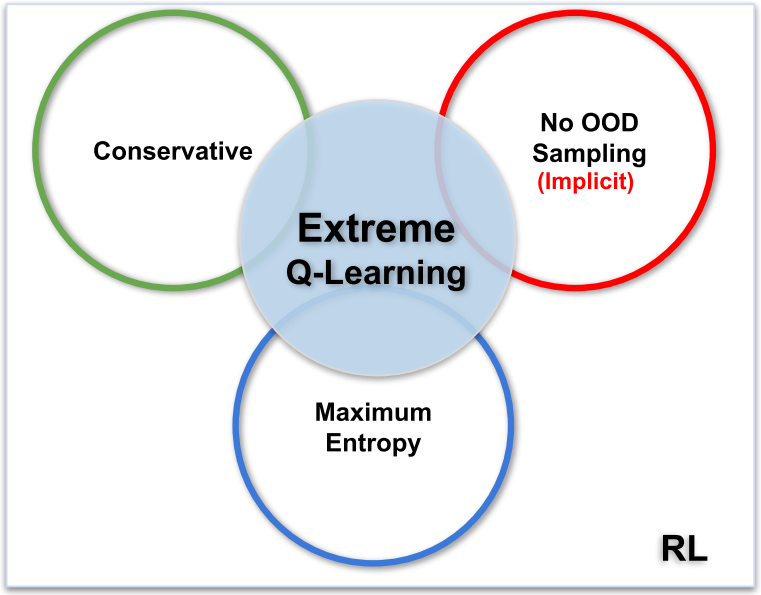
Andrew Barto and Richard Sutton Receive 2024 ACM A.M. Turing Award

Andrew G. Barto and Richard S. Sutton received the 2024 ACM A.M. Turing Award for developing the conceptual and algorithmic foundations of reinforcement learning. In a series of papers beginning in the 1980s, Barto and Sutton introduced the main ideas, constructed the mathematical foundations, and developed important algorithms for reinforcement learning—one of the most important approaches for creating intelligent systems. Barto is Professor Emeritus of Information and Computer Sciences at the University of Massachusetts, Amherst. Sutton is a Professor of Computer Science at the University of Alberta, a Research Scientist at Keen Technologies, and a Fellow at Amii (Alberta Machine Intelligence Institute).

https://www.acm.org/media-center/2025/march/turing-award-2024

In Short

- 최대 엔트로피(MaxEnt) RL의 한계를 극복하기 위한 새로운 Extreme Q-Learning 프레임워크 개발
- Extreme Value Theory (EVT)와 Gumbel regression을 활용하여 최적의 소프트 값(soft value)을 policy network로부터의 샘플링 없이(그러므로 Out-of-distribution 문제가 없음) 직접 추정하는 방법을 제안
- Key Idea: Use the Gumbel distribution to fit the soft-optimal value function



Extreme Q-Learning Algorithm

1. Learn V^* via $\mathcal{L}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{\left(\hat{Q}^k(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[\left(\hat{Q}^k(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta\right] - 1$

2. Update Q via $\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})}\left[V^*(\mathbf{s}')\right]$

3. Repeat 1-2

Finally, $\pi^* = \arg\min_{\pi} \mathbb{E}_{\rho_\mu(\mathbf{s},\mathbf{a})}\left[e^{\left(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta}\log\pi\right]$

발표 개요

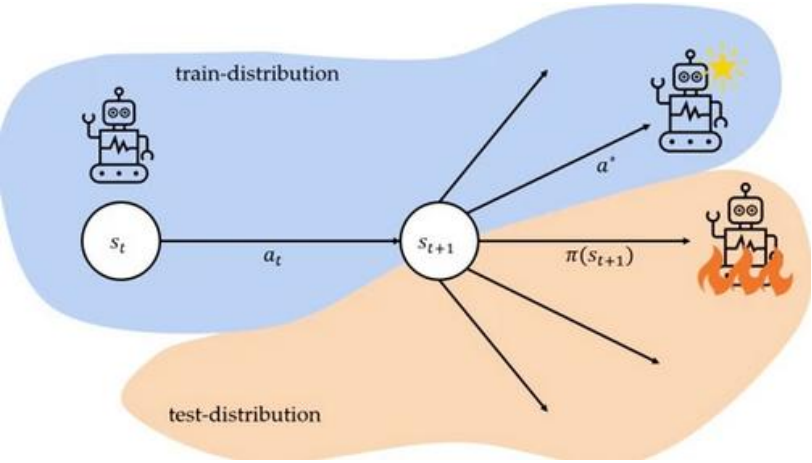
- 배경 및 동기
  - 전통 Q-러닝과 MaxEnt RL의 문제점
- 최대 엔트로피 RL 개요
  - 소프트 벨만 연산자(soft-Bellman operator)와 log-sum-exp 문제
- 극단값 이론 (EVT)와 Gumbel 분포
  - EVT 기본 개념 및 Gumbel-Max Trick
- Gumbel Regression를 통한 소프트 값 추정
  - 수학적 유도 및 직관
- Extreme Q-Learning 알고리즘
  - 온라인 및 오프라인 업데이트 방식
- 실험 결과 및 논의
- 결론 및 미래 연구 방향

What is the Problem?

- Optimal Bellman operator는 max 연산을 사용하는데 연속 행동 공간에서는 intractable함

$$\mathrm{T}^{*}Q = r\left(s_t, a_t\right) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right)$$

- 이 문제를 해결하기 위한 기존 방법들
  - Actor-Critic 방식
    - 정책 함수 $\pi(a|s)$를 학습하여 최적 행동을 직접 찾도록 설계
    - Critic 네트워크 $Q(s,a)$가 주어진 상태에서 Q값을 평가
    - Actor 네트워크 $\pi(a|s)$가 최적 행동을 직접 출력하여 $\max_{a'} Q(s',a')$를 근사
    - Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC)
  - 정책 최적화 기반 접근 (Policy Optimization)
    - Q-learning처럼 Q값을 찾지 않고, 정책을 직접 최적화
    - Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO)
  - out-of-distribution 문제가 발생할 수 있음



- 목표
  - 정책 네트워크 없이 Q함수의 최대값(Soft Bellman Backup)을 직접 추정할 수 있는 방법 제시

What was Maximum Entropy RL?

- MaxEnt RL의 목표
  - 보상 최대화와 함께 정책의 엔트로피(불확실성)를 증가시킴

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_t r_t + \alpha \mathcal{H}\left(\pi\left(\cdot|s_t\right)\right)\right]$$

- optimal maximum entropy bellman equation can be rewritten with log-sum-exp (details later)

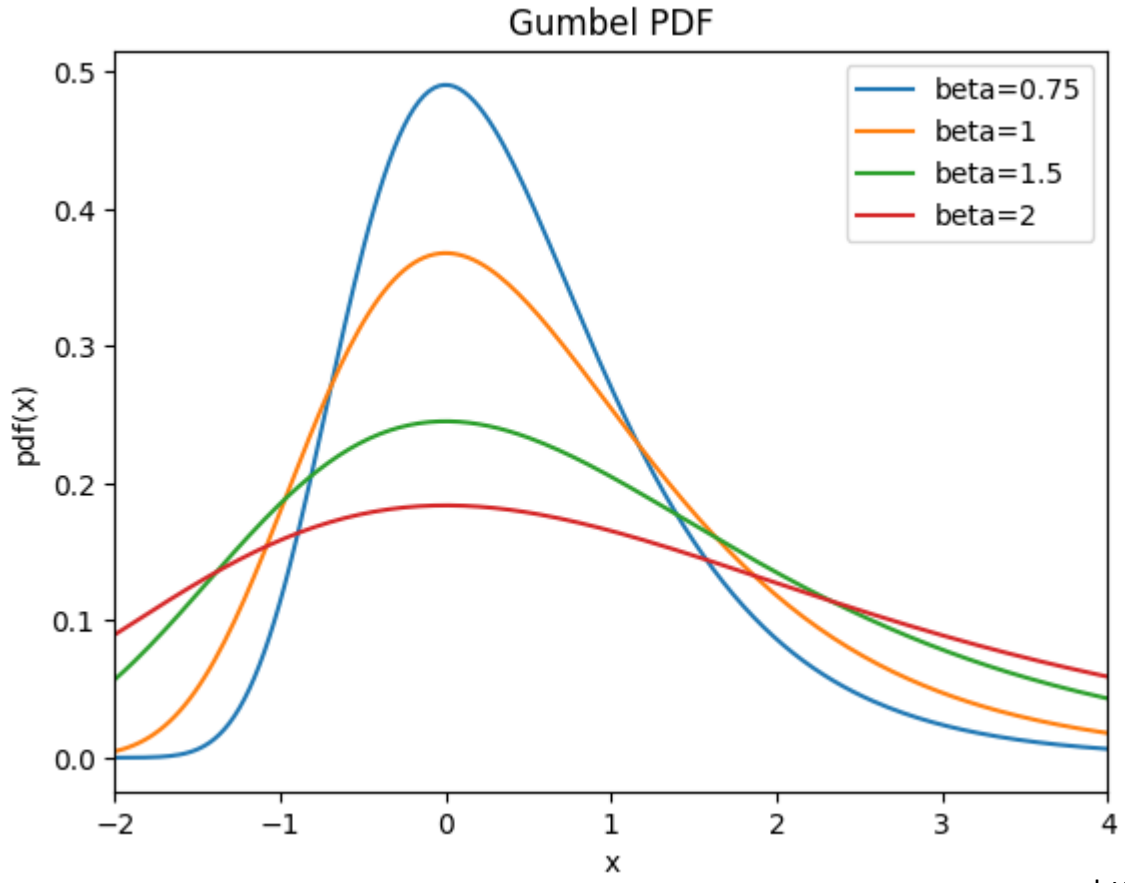
$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \underbrace{\log \mathbb{E}_a\left[e^{Q^*(s_{t+1}, a_{t+1})}\right]}_{\text{still intractable!}}$$

- 본 연구의 목표: log-sum-exp를 직접 다뤄보자!
  - 어떻게?
  - 노벨경제학상을 받은 연구 중 유용한 연구가 있다 (McFadden's 2000 Nobel-prize winning work in Economics)
  - 이 연구는 extreme value theory를 활용했다
  - soft-optimal utility functions with logit (or softmax) choice probability naturally arise when utilities are assumed to have Gumbel-distributed errors

Gumbel Distribution

- PDF of Gumbel distribution $\mathcal{G}(\mu, \beta)$

$$f_X(x) = \frac{1}{\beta} e^{-\left(\frac{x-\mu}{\beta}\right) - e^{-\left(\frac{x-\mu}{\beta}\right)}} = \frac{1}{\beta} \exp\left(-\frac{x-\mu}{\beta} - \exp\left(-\frac{x-\mu}{\beta}\right)\right)$$



https://en.wikipedia.org/wiki/Gumbel_distribution

Gumbel Distribution and RL

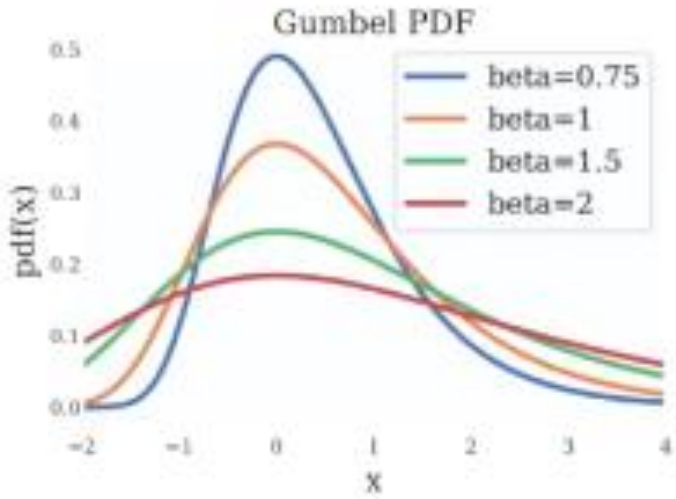
$$\text{Target } y = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1})$$

$$= r(s_t, a_t) + \gamma \max_{a_{t+1}} \left(r(s_{t+1}, a_{t+1}) + \gamma \max_{a_{t+2}} (Q^{\pi}(s_{t+2}, a_{t+2}) + \epsilon_{t+2}) + \epsilon_{t+1} \right)$$

...

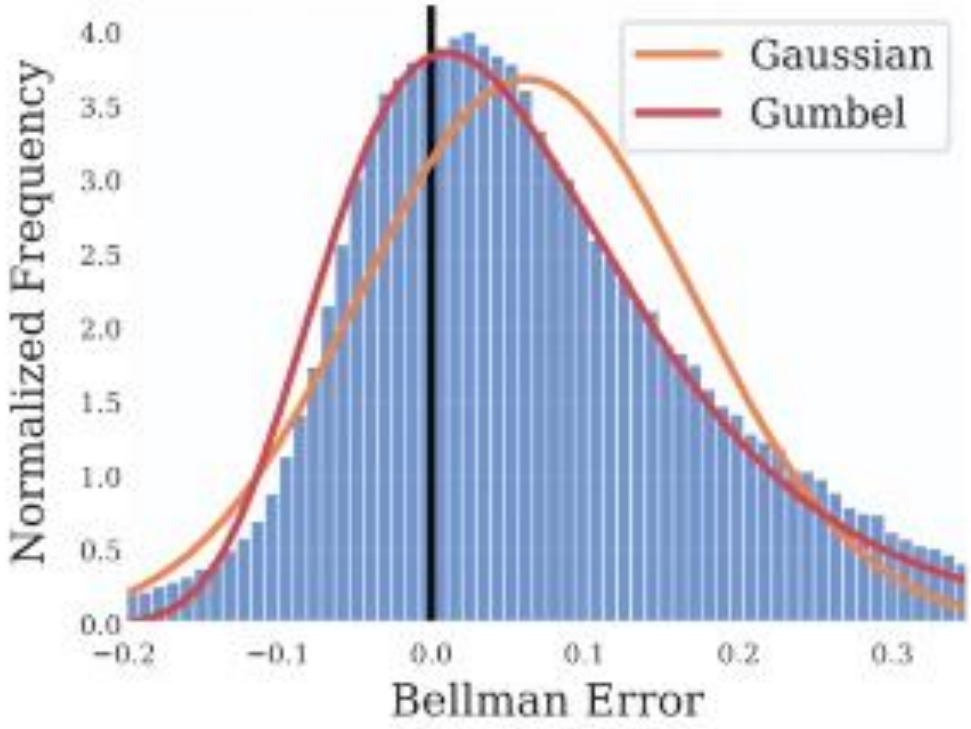
$$= \text{target} + \max(\epsilon_{t+1} + \epsilon_{t+2} + \epsilon_{t+3} + \cdots) \quad \Rightarrow \quad \text{EVT}$$

Max errors tend to aggregate!!



Bellman Update Errors from SAC on Half-cheetah

- Errors are Gumbel distributed, not Gaussian!
  - Q함수 근사 오차가 Gumbel 분포를 따를 경우, max 연산이 log-sum-exp 연산과 동일하게 작용할 수 있음



Log-Sum-Exp Expression

- max 연산은 discontinuity를 야기하므로 최적화가 어렵거나 비선형성이 강해지는 문제가 발생
  - 이를 완화하기 위해 log-sum-exp (LSE)를 활용하여 다음과 같이 변환할 수 있음

$$\max\left(x_1, x_2, \ldots, x_n\right) \approx \ln \sum_i \exp\left(x_i\right)$$

- 근사 정확도 조절
  - 스케일링 파라미터 β를 추가하여 근사 정확도를 조절할 수 있음

$$\max\left(x_1, x_2, \ldots, x_n\right) \approx \beta \ln \sum_i \exp\left(x_i / \beta\right)$$

    - $\beta \to 0$이면 정확한 max 함수로 수렴
    - $\beta \to \infty$이면 모든 값의 산술평균과 유사하게 됨

- 위의 변환을 max 함수의 smooth approximation이라고 부름
  - log-sum-exp 변형을 사용하면 max를 부드럽게 근사할 수 있으며, tractable하게 변환 가능

Log-Partition Function (1)

- log-partition function
  - log-partition function은 확률 분포의 정규화 상수를 로그 스케일로 표현한 것이며, log-sum-exp와 수학적으로 동일한 형태를 가짐
  - 아래와 같은 Partition function이 있을 때

$$Z(\theta) = \sum_{x} \exp\left(f(x,\theta)\right)$$

  - 다음과 같이 확률분포를 정의하면 아래 확률분포는 Boltzmann 분포 또는 Gibbs 분포의 형태가 됨

$$p_\theta(x) = \frac{\exp\left(f(x,\theta)\right)}{Z(\theta)}$$

  - 이 분포를 이용하여 다시 표현하면

$$Z(\theta) = \sum_{x} \frac{\exp\left(f(x,\theta)\right)}{p_\theta(x)} p_\theta(x) = \mathbb{E}_{p_\theta(x)}\left[\frac{\exp\left(f(x,\theta)\right)}{p_\theta(x)}\right]$$

  - 위의 식에 log를 취하고 Jensen's inequality를 적용하면

$$A(\theta) = \ln Z(\theta) = \ln \mathbb{E}_{p_\theta(x)}\left[\frac{\exp\left(f(x,\theta)\right)}{p_\theta(x)}\right] \geq \mathbb{E}_{p_\theta(x)}\left[\ln \frac{\exp\left(f(x,\theta)\right)}{p_\theta(x)}\right]$$

Log-Partition Function (2)

- log-partition function
  - 이 때, Jensen's inequality의 부등호를 등호로 바꿀 수 있는데 $Z(\theta) = \frac{\exp(f(x,\theta))}{p_\theta(x)}$ 이므로 log의 인자가 x에 의존하지 않는 상수이기 때문임
  - 즉, 모든 x에 대해 동일한 값을 가지므로 상수 조건을 만족

$$A(\theta) = \ln \mathbb{E}_{p_\theta(x)}\left[\frac{\exp(f(x,\theta))}{p_\theta(x)}\right] = \mathbb{E}_{p_\theta(x)}\left[\ln \frac{\exp(f(x,\theta))}{p_\theta(x)}\right]$$

$$= \mathbb{E}_{p_\theta(x)}[f(x,\theta)] \underbrace{-\mathbb{E}_{p_\theta(x)}[\ln p_\theta(x)]}_{\text{entropy term}}$$

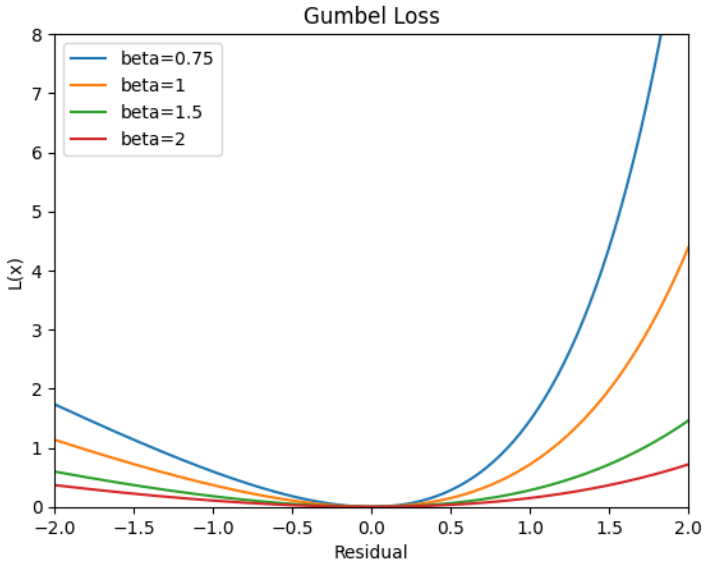
  - 이처럼 log-partition function은 entropy를 내재하기 때문에 implicit하게 entropy를 고려할 수 있음

Log-sum-exp Approximation via Gumbel Regression (1)

- Gumbel Regression fits a Gumbel distribution $\mathcal{G}(\mu, \beta)$ using Maximum Likelihood Estimation (MLE) to the estimation errors for learning a model $\hat{y}$ using samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- This gives the Gumbel Regression Loss:

$$L(y) = \exp\left((y - \hat{y})/\beta\right) - (y - \hat{y})/\beta - 1$$

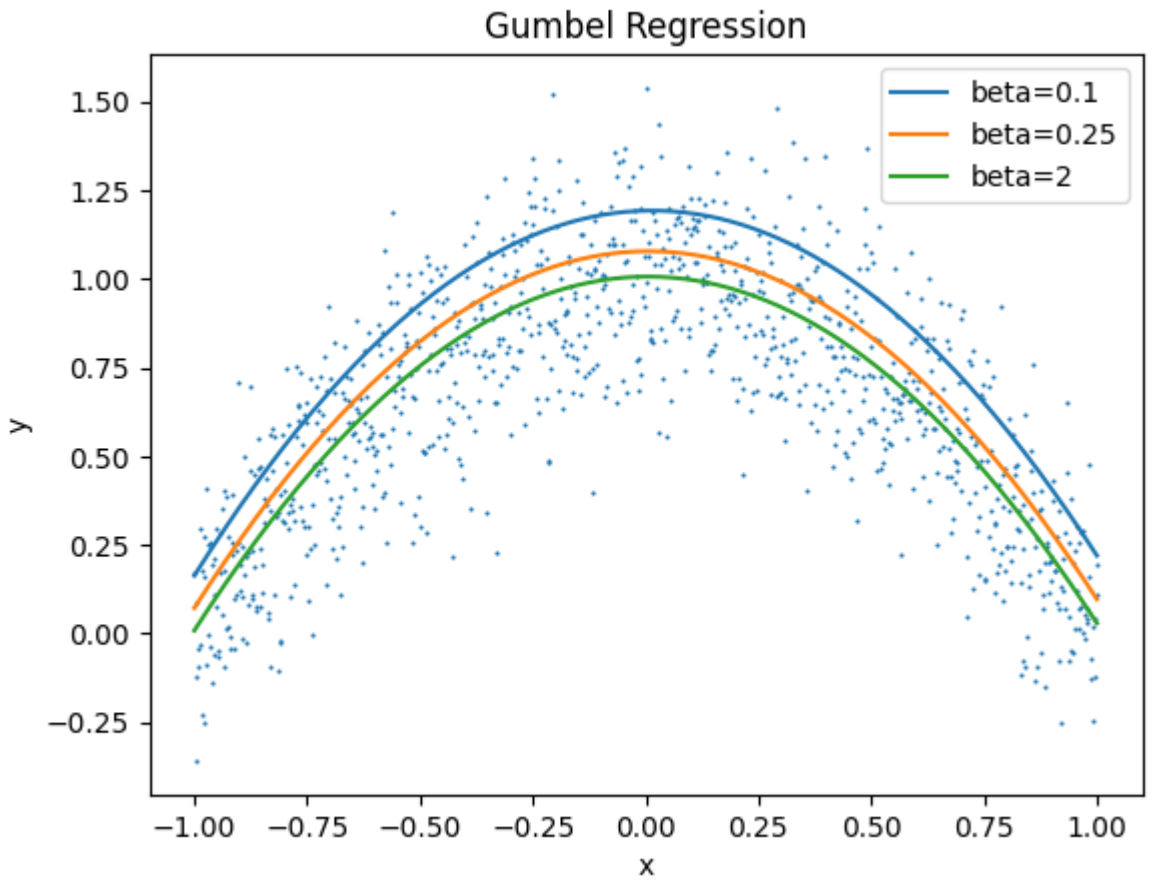
- More formally Gumbel regression is
  - fitting Gumbel distribution $\mathcal{G}(\mu, \beta)$ to the data to model the extreme values of a distribution
  - For a temperature β, Gumbel regression estimates the smooth approximation of the max operator, given by $\beta \log \mathbb{E}\left[e^{X/\beta}\right]$, which is related to the Log-Partition function over samples drawn from a distribution X
- Gumbel Regression enables exact estimation of the Log-Partition function by using simple gradient descent



Log-sum-exp Approximation via Gumbel Regression (2)

- A simple example of Gumbel Regression
  - Here we look at a simple regression example for a quadratic with some gaussian noise $y = -x^2 + 1 + \mathcal{N}\left(0, 0.2^2\right)$

For low temps, this fits the maximal values of the data and for high temps it will fit the mean values of the data



This is similar to fitting a Gaussian distribution to the data, but instead of modeling the mean (i.e. least squares regression) it fits the Log-Sum-Exp or the Log-Partition function of the data

Optimal Solution for MaxEnt RL

- MaxEnt RL with reference policy μ

$$\max_{\pi} E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\left(r\left(s_{t},a_{t}\right)+\beta\log\frac{\pi\left(a_{t}|s_{t}\right)}{\mu\left(a_{t}|s_{t}\right)}\right)\right]$$

KL Divergence with reference distribution $D_{\mathrm{KL}}(\pi\|\mu)$

- Optimal solution for MaxEnt RL with soft value function

$$Q^{*}\left(s_{t},a_{t}\right)=r\left(s_{t},a_{t}\right)+\gamma E_{s_{t+1}}\left[V^{*}\left(s_{t+1}\right)\right]$$

- soft value function
  - log-sum-exp shape!

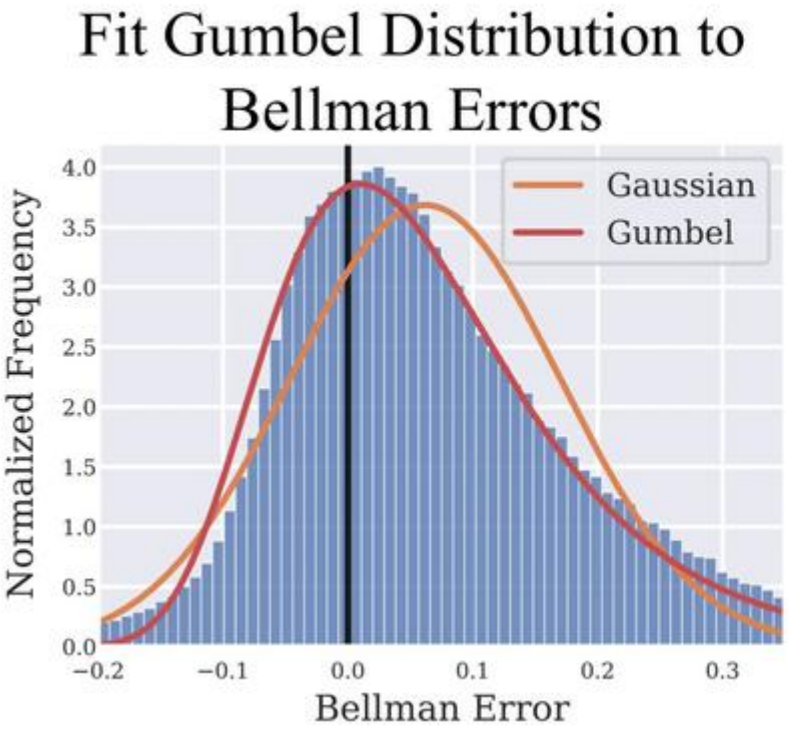
$$V^{*}\left(s_{t}\right)=\beta\log\mathbb{E}_{a_{t}\sim\mu\left(\cdot|s_{t}\right)}\left[\exp\left(Q^{*}\left(s_{t},a_{t}\right)/\beta\right)\right]$$

Extreme Q-Learning

- Gumbel regression loss function can be used to directly fit the Log-Sum-Exp of the Q-values
- Then, we get the soft-optimal value function

$$V^*(s) = \beta \log \mathbb{E}_{a\sim\mu}\left[\exp\left(Q(s,a)/\beta\right)\right]$$

- Then, we can use Q-iteration even in high-dimensional continuous action spaces to find the optimal MaxEnt policy. This general algorithm works well in both online, and offline settings



Gumbel Regression Loss

$$\mathcal{L}(h) = \mathbb{E}_{x\sim\mu}\left[e^{(x-h)/\beta} - (x-h)/\beta - 1\right]$$

convex loss function with unique minima

$$h^* = \beta \log \mathbb{E}_{x\sim\mu}[e^{x/\beta}]$$

aka the *LogSumExp*!

Extreme Q-Learning (2)

Extreme Q - Learning Algorithm

1. Learn V^* via $\mathcal{L}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{\left(\hat{Q}^k(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[\left(\hat{Q}^k(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta\right] - 1$

2. Update Q via $\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})}\left[V^*(\mathbf{s}')\right]$, $\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'}\left[\left(Q(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a}) - V^*(\mathbf{s}')\right)^2\right]$

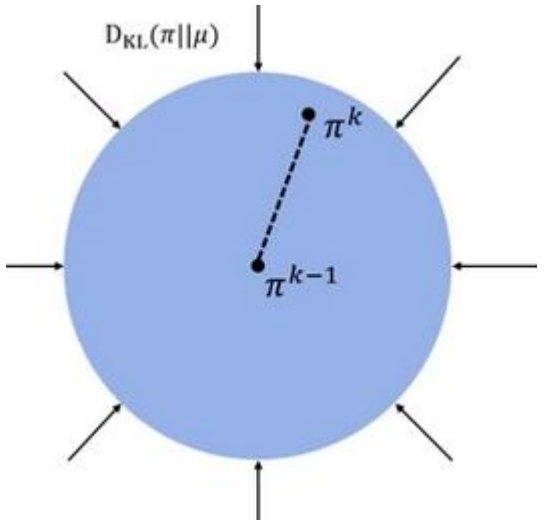
3. Repeat 1-2

Finally, $\pi^* = \arg\min_{\pi} \mathbb{E}_{\rho_\mu(\mathbf{s},\mathbf{a})}\left[e^{\left(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s})\right)/\beta}\log\pi\right]$

Online RL Evaluation을 위한 구현

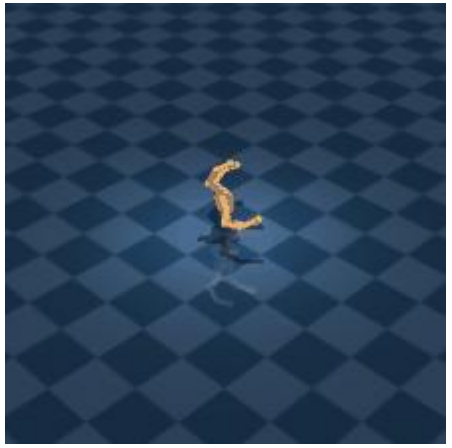
- online RL
  - Choose $\mu = \pi^{k-1}$
  - Run $\mathcal{X}$QL
    - 1) $\mathcal{L}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta} - \frac{Q(\mathbf{s},\mathbf{a})-V(\mathbf{s})}{\beta} - 1 \right]$
    - 2) $\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'}\left[\left(Q(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a}) - V^*(\mathbf{s}') \right)^2 \right]$
  - Policy extraction via min $\boldsymbol{D_{\mathrm{KL}}(\pi\|\mu)}$:

$$\max_{\pi} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi^{k-1}}\left[Q^*(\mathbf{s},\pi(\mathbf{s})) \right]$$



Evaluations (1)

- Online RL Results



< X-TD3 on Quadruped Run (Reward 437) >

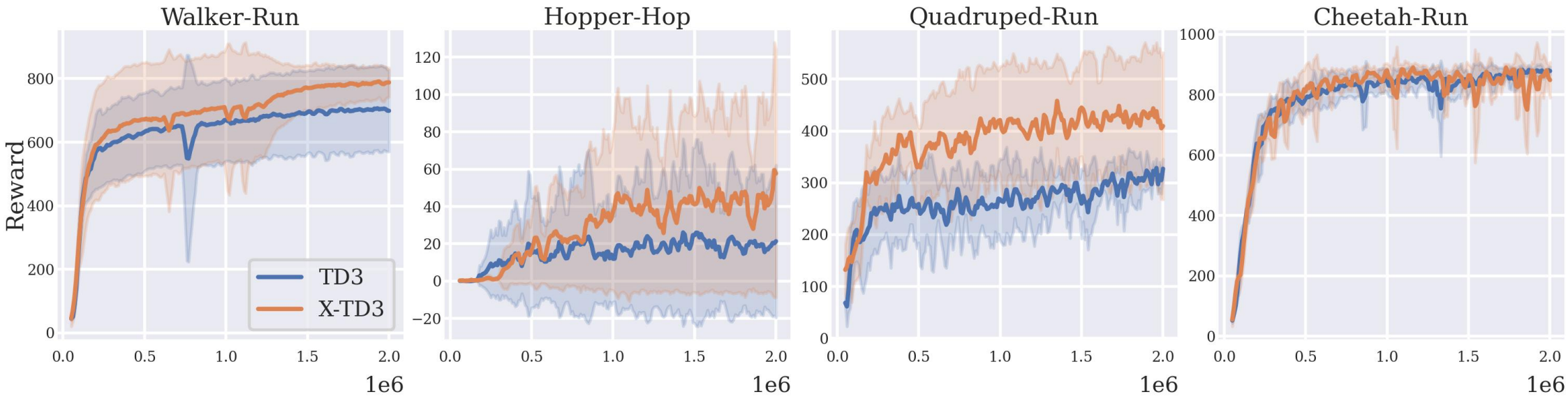
< TD3 on Quadruped Run (Reward 293) >

< X-TD3 on Hopper Hop (Reward 71) >

< TD3 on Hopper Hop (Reward 20) >

Evaluations (2)

- Online RL Results
  - DeepMind Control 벤치마크에서 SAC, TD3와 비교해 비슷하거나 약간 개선된 성능

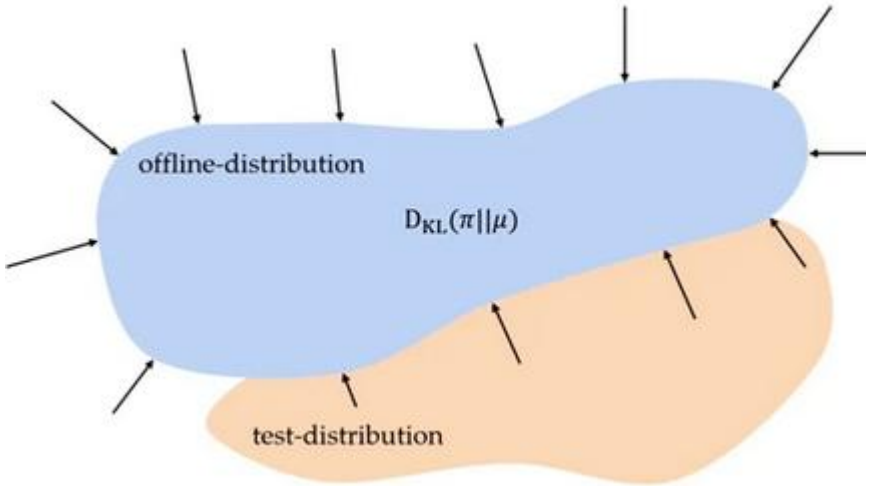


< X-TD3 shows moderate gains on DM Control Tasks compared to standard TD3 >

Offline RL Evaluation을 위한 구현

- offline RL
  - Choose $\mu = \mathcal{D}$
  - Run $\mathcal{X}$QL
    - 1) $\mathcal{L}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta} - \frac{Q(\mathbf{s},\mathbf{a})-V(\mathbf{s})}{\beta} - 1 \right]$
    - 2) $\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'}\left[\left(Q(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a}) - V^*(\mathbf{s}') \right)^2 \right]$
  - Policy extraction via min $\boldsymbol{D_{\mathrm{KL}}(\pi\|\mu)}$:
    - 정책은 학습 후 AWR objective를 통해 추출

$$\max_{\pi} \mathbb{E}_{(s,a)\sim D}\left[e^{\left(Q^*(s,a)-V^*(s)\right)/\beta} \log \pi(a|s) \right]$$

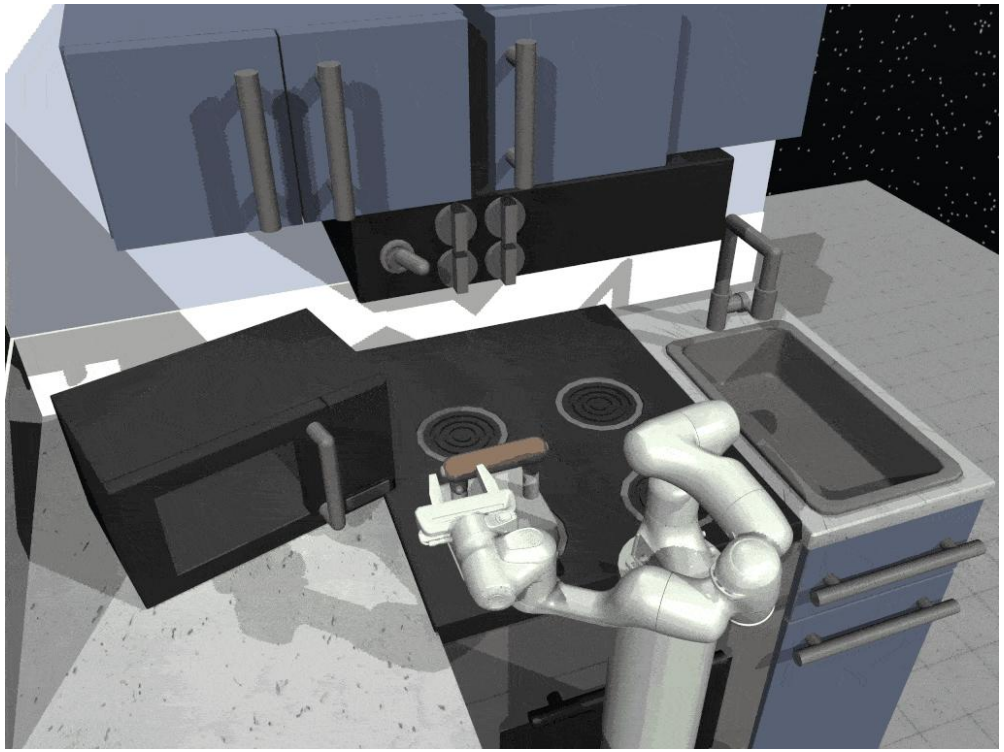


Evaluations (3)

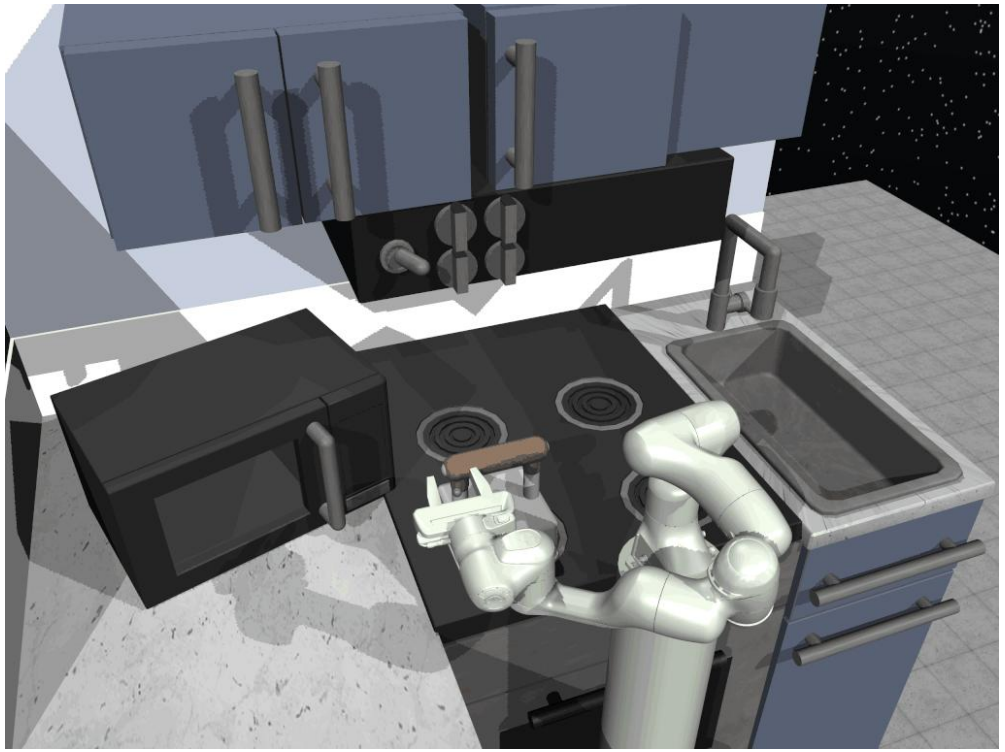
- Offline RL Results
  - D4RL 벤치마크 (Franka Kitchen, MuJoCo, AntMaze 등)에서 기존 CQL, IQL 대비 우수한 성능

Evaluations (4)

- Offline RL Results
  - D4RL 벤치마크 (Franka Kitchen, MuJoCo, AntMaze 등)에서 기존 CQL, IQL 대비 우수한 성능



< XQL on Franka Kitchen >



< IQL on Franka Kitchen >

장점 및 개선사항

- 주요 장점
  - 명시적 엔트로피 추정 없이도 소프트 밸류 근사
  - 연속 행동 공간에서의 계산 효율성 및 안정성 향상
  - 오프라인 RL에서의 out-of-distribution 문제 최소화

- 개선해야할 부분
  - β 민감성 해결을 위한 하이퍼파라미터 자동 조정 가능성
  - 더 복잡한 환경 및 실제 문제 적용

$$\mathcal{L}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta} - \frac{Q(\mathbf{s},\mathbf{a})-V(\mathbf{s})}{\beta} - 1 \right]$$

Key takeaway: EVT와 Gumbel 회귀를 활용해 기존 MaxEnt RL의 한계를 극복한 새로운 Extreme Q-Learning 프레임워크 제시

Proof of LSE $\beta \ln \sum_i \exp\left(\frac{x_i}{\beta}\right)$ Approximation

- $\beta \to 0$이면 LSE는 max 함수로 수렴함을 증명
  - 최대값 $x_{\max} = \max\{x_i\}$에 대해

$$\sum_i \exp\left(\frac{x_i}{\beta}\right) = \exp\left(\frac{x_{\max}}{\beta}\right)\left(1 + \sum_{j \neq \max} \exp\left(\frac{1}{\beta}(x_i - x_{\max})\right)\right)$$

  - 여기서 $x_j - x_{\max} < 0$이므로, $\exp\left((x_i - x_{\max})/\beta\right)$는 β가 증가할수록 0에 가까워짐

  - 즉, 합 전체가 $\exp\left(\frac{x_{\max}}{\beta}\right)$에 의해 지배됨

$$\beta \ln \sum_i \exp\left(\frac{x_i}{\beta}\right) \approx \beta \ln \exp\left(\frac{x_{\max}}{\beta}\right) = \beta \frac{x_{\max}}{\beta} = x_{\max}$$

  - 따라서 $\beta \to 0$이면 LSE는 max 함수로 수렴함

Proof of LSE $\beta \ln \sum_i \exp\left(\frac{x_i}{\beta}\right)$ Approximation

- $\beta \to \infty$이면 LSE는 평균으로 근사함을 증명
  - 테일러 전개를 하면

$$\exp\left(\frac{x_i}{\beta}\right) \approx 1 + \frac{x_i}{\beta}$$

$$\sum_i \exp\left(\frac{x_i}{\beta}\right) \approx \sum_i \left(1 + \frac{x_i}{\beta}\right) = n + \frac{1}{\beta}\sum_i x_i$$

  - 로그를 취하면

$$\ln \sum_i \exp\left(\frac{x_i}{\beta}\right) \approx \ln\left(n + \frac{1}{\beta}\sum_i x_i\right)$$

  - $\beta \to \infty$일 때, $\beta \ln(1+\epsilon) \approx \epsilon$ 근사를 이용하면

$$\ln \sum_i \exp\left(\frac{x_i}{\beta}\right) \approx \ln n\left(1 + \frac{1}{\beta n}\sum_i x_i\right) = \ln n + \ln\left(1 + \frac{1}{\beta n}\sum_i x_i\right) \approx \ln n + \frac{1}{\beta n}\sum_i x_i$$

  - β를 곱하고, $\beta \to \infty$이면 $\beta \ln n \to 0$이므로 $\beta \to \infty$이면 LSE는 산술평균에 근사함

$$\beta \ln \sum_i \exp\left(\frac{x_i}{\beta}\right) \approx \beta \ln n + \frac{1}{n}\sum_i x_i \approx \frac{1}{n}\sum_i x_i$$

Numerical Example for Log-Partition Function

```python
import numpy as np

# Define f(x, theta) as some arbitrary function
theta = 1.5
x_values = np.array([1, 2, 3, 4, 5])
f_x_theta = theta * x_values  # Example function f(x, theta) = theta * x

# Compute partition function Z(theta)
Z_theta = np.sum(np.exp(f_x_theta))

# Compute probability distribution p_theta(x)
p_theta_x = np.exp(f_x_theta) / Z_theta

# Compute expectation of log(exp(f(x, theta)) / p_theta(x))
expected_value = np.sum(p_theta_x * np.log(np.exp(f_x_theta) / p_theta_x))

# Compute log-partition function directly
log_partition = np.log(Z_theta)

# Display results in a simple format
print("Numerical Example for Log-Partition Function")
print("="*50)
print(f"{'x':<10}{'f(x, theta)':<15}{'exp(f(x, theta))':<20}{'p_theta(x)':<20}{'log(exp(f(x, theta)) / p_theta(x))':<30}")
for i in range(len(x_values)):
    print(f"{x_values[i]:<10}{f_x_theta[i]:<15.5f}{np.exp(f_x_theta[i]):<20.5f}{p_theta_x[i]:<20.5f}{np.log(np.exp(f_x_theta[i]) / p_theta_x[i]):<30.5f}")

# Show numerical verification
print("\nNumerical Verification:")
print(f"Expected Value: {expected_value:.5f}")
print(f"Log-Partition Function: {log_partition:.5f}")
```

```
Numerical Example for Log-Partition Function
==================================================
x         f(x, theta)     exp(f(x, theta))    p_theta(x)          log(exp(f(x, theta)) / p_theta(x))
1         1.50000         4.48169             0.00193             7.75193
2         3.00000         20.08554            0.00864             7.75193
3         4.50000         90.01713            0.03870             7.75193
4         6.00000         403.42879           0.17344             7.75193
5         7.50000         1808.04241          0.77730             7.75193

Numerical Verification:
Expected Value: 7.75193
Log-Partition Function: 7.75193
```