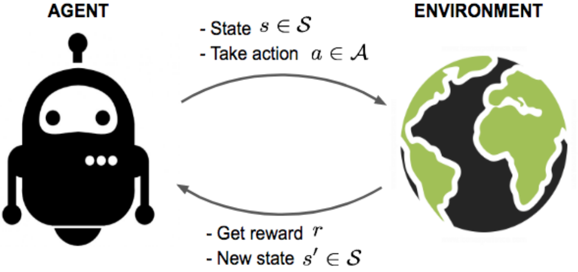


Final Report

Kushagra Chandak
International Institute of Information Technology, Hyderabad (IIIT-H), India

Where do I come from?

Where do I come from?

Where do I come from?

Contents

Introduction

Policy Gradient

- Policy based RL: $\pi_\theta(s, a) = P(a|s, \theta)$

Policy Gradient

- Policy based RL: $\pi_\theta(s, a) = P(a|s, \theta)$
- $a = \mu(s)$ vs $\pi(a, s) = P(a|s)$

Policy Gradient

- Policy based RL: $\pi_\theta(s, a) = P(a|s, \theta)$
- $a = \mu(s)$ vs $\pi(a, s) = P(a|s)$
- Policy objective functions: $J(\theta) = \mathbb{E}_{\mu'}\left[\sum_t r(s_t, a_t)|\pi_\theta(s, a)\right]$

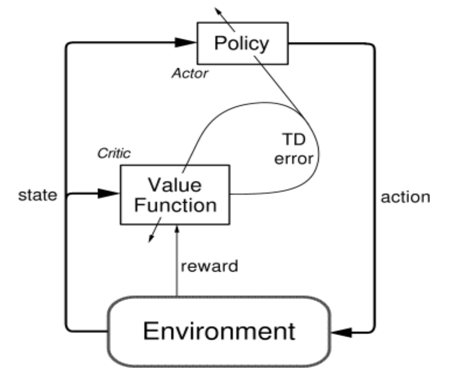
Policy Gradient

- Policy based RL: $\pi_\theta(s, a) = P(a|s, \theta)$
- $a = \mu(s)$ vs $\pi(a, s) = P(a|s)$
- Policy objective functions: $J(\theta) = \mathbb{E}_{\mu'}\left[\sum_t r(s_t, a_t)|\pi_\theta(s, a)\right]$
- Optimize policy by computing noisy estimates of ∇J and then updating policy in gradient's direction.

Policy Gradient

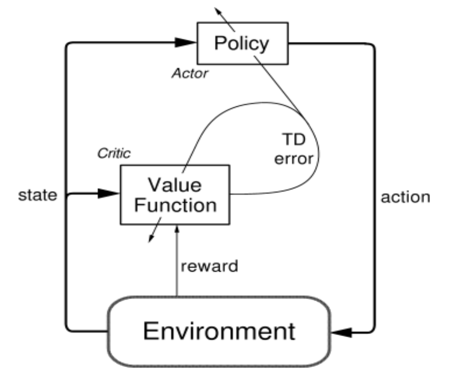
- Policy based RL: $\pi_\theta(s, a) = P(a|s, \theta)$
- $a = \mu(s)$ vs $\pi(a, s) = P(a|s)$
- Policy objective functions: $J(\theta) = \mathbb{E}_{\mu'}\left[\sum_t r(s_t, a_t)|\pi_\theta(s, a)\right]$
- Optimize policy by computing noisy estimates of ∇J and then updating policy in gradient's direction.
- A lot of problems with vanilla PG (continuous domains, credit assignment, etc.)

Actor-Critic



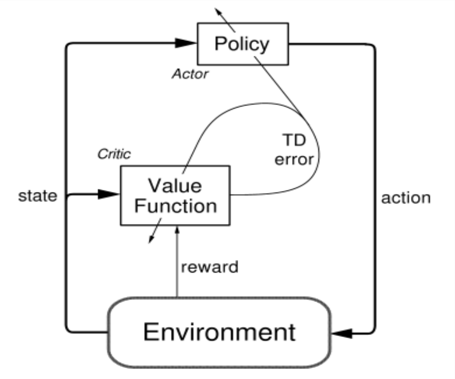
- Represent policy *independent* of the value function.

Actor-Critic



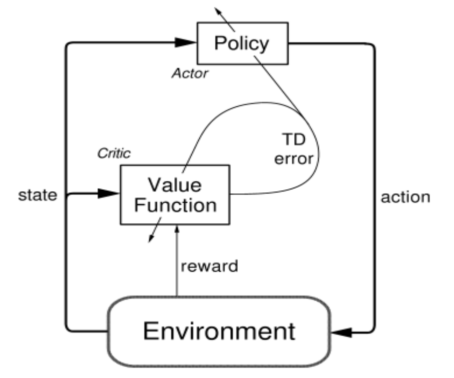
- Represent policy *independent* of the value function.
- Policy: Actor, Value: Critic

Actor-Critic



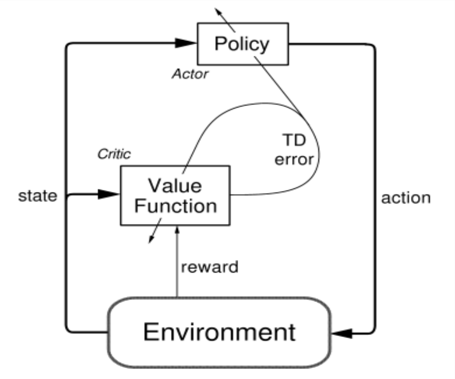
- Represent policy *independent* of the value function.
- Policy: Actor, Value: Critic
- Given current state, actor produces an action.

Actor-Critic



- Represent policy *independent* of the value function.
- Policy: Actor, Value: Critic
- Given current state, actor produces an action.
- Critic produces error signal given state and reward. It's output drives learning in both actor and critic.

Actor-Critic



- Represent policy *independent* of the value function.
- Policy: Actor, Value: Critic
- Given current state, actor produces an action.
- Critic produces error signal given state and reward. It's output drives learning in both actor and critic.
- In DRL, neural nets can be used to represent actor and critic.

Baseline Algorithms

Deep Deterministic Policy Gradient (DDPG)[1]

- Motivation: To tackle continuous action spaces.

[1] Lillicrap et al., 2015

Deep Deterministic Policy Gradient (DDPG)[1]

- Motivation: To tackle continuous action spaces.
- Combines *three* techniques together:

[1] Lillicrap et al., 2015

Deep Deterministic Policy Gradient (DDPG)[1]

- Motivation: To tackle continuous action spaces.
- Combines *three* techniques together:
  - Deterministic Policy Gradient, Actor Critic and Deep Q-Network

[1] Lillicrap et al., 2015

Deep Deterministic Policy Gradient (DDPG)[1]

- Motivation: To tackle continuous action spaces.
- Combines *three* techniques together:
  - Deterministic Policy Gradient, Actor Critic and Deep Q-Network
- DQN: Learning is stabilized using experience replay and frozen target network.

[1] Lillicrap et al., 2015

Deep Deterministic Policy Gradient (DDPG)[1]

- Motivation: To tackle continuous action spaces.
- Combines *three* techniques together:
    - Deterministic Policy Gradient, Actor Critic and Deep Q-Network
- DQN: Learning is stabilized using experience replay and frozen target network.
- DDPG: Extends DQN to continuous space with actor-critic framework while learning a deterministic policy.

[1] Lillicrap et al., 2015

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic:

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $\mu(s_{t+1})$

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $Q(s_{t+1}, \mu(s_{t+1}))$

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$
- To train, use s_t, a_t as inputs and y_t as target.

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$
- To train, use s_t, a_t as inputs and y_t as target.
- Training actor: Deterministic Policy Gradient!

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$
- To train, use s_t, a_t as inputs and y_t as target.
- Training actor: Deterministic Policy Gradient!
- $\nabla_{\theta^\mu} J = \mathbb{E}_{\mu'} \left[\nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu) \right]$

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$
- To train, use s_t, a_t as inputs and y_t as target.
- Training actor: Deterministic Policy Gradient!
- $\nabla_{\theta^\mu} J = \mathbb{E}_{\mu'} \left[\nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu) \right]$
- Use actor's and critic's online network for $\mu(s)$ and ∇Q respectively.

DDPG: The algorithm

- Choosing action: Get $a = \mu(s)$ from actor. While training, add exploration noise.
- Training critic: $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))$
- To train, use s_t, a_t as inputs and y_t as target.
- Training actor: Deterministic Policy Gradient!
- $\nabla_{\theta^\mu} J = \mathbb{E}_{\mu'} \left[\nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu) \right]$
- Use actor's and critic's online network for $\mu(s)$ and ∇Q respectively.
- Calculate $\nabla \mu(s)$ given ∇Q and apply those gradient to actor's net.

DDPG: What's inside?

- Soft updates (conservative policy iteration) on the parameters of both actor and critic, with update rate $\tau << 1$:

$$\theta^{'} \leftarrow \tau\theta + (1 - \tau)\theta^{'}$$

DDPG: What's inside?

- Soft updates (conservative policy iteration) on the parameters of both actor and critic, with update rate $\tau << 1$:

$$\theta^{'} \leftarrow \tau\theta + (1 - \tau)\theta^{'}$$

- The target network values change slowly, unlike DQN where it's frozen for sometime.

DDPG: What's inside?

- Soft updates (conservative policy iteration) on the parameters of both actor and critic, with update rate $\tau << 1$:

$$\theta^{'} \leftarrow \tau\theta + (1 - \tau)\theta^{'}$$

- The target network values change slowly, unlike DQN where it's frozen for sometime.
- Batch Normalization: Normalizing every dimension across samples in one mini batch. (Not required for simple tasks)

DDPG: What's inside?

- Soft updates (conservative policy iteration) on the parameters of both actor and critic, with update rate $\tau << 1$:

$$\theta^{'} \leftarrow \tau\theta + (1-\tau)\theta^{'}$$

- The target network values change slowly, unlike DQN where it's frozen for sometime.
- Batch Normalization: Normalizing every dimension across samples in one mini batch. (Not required for simple tasks)
- Better exploration: An exploration policy is $\mu^{'}$ is constructed by adding noise $\mathcal{N}$

$$\mu^{'}(s) = \mu_{\theta}(s) + \mathcal{N}$$

Ornstein-Uhlenbeck (OU) Process

- Used for modeling biological processes such as neuronal response, and in mathematical finance.

Ornstein-Uhlenbeck (OU) Process

- Used for modeling biological processes such as neuronal response, and in mathematical finance.
- A stochastic process: $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$

Ornstein-Uhlenbeck (OU) Process

- Used for modeling biological processes such as neuronal response, and in mathematical finance.
- A stochastic process: $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$
- Discretized form: $x_{n+1} = x_n + \theta(\mu - x_n)\Delta t + \sigma \Delta W_n$

Ornstein-Uhlenbeck (OU) Process

- Used for modeling biological processes such as neuronal response, and in mathematical finance.
- A stochastic process: $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$
- Discretized form: $x_{n+1} = x_n + \theta(\mu - x_n)\Delta t + \sigma \Delta W_n$
- W_t: Brownian motion, $\Delta W_n \sim N(0, \Delta t) = \sqrt{\Delta t} N(0, 1)$

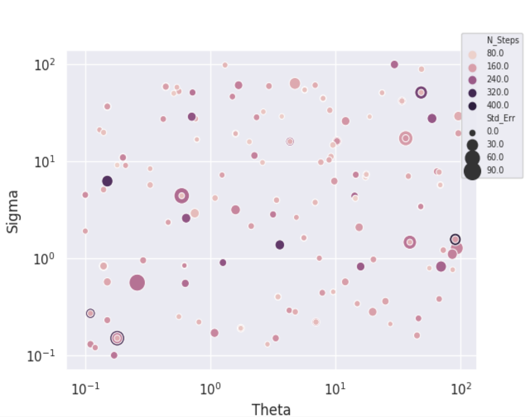
Ornstein-Uhlenbeck (OU) Process

- Used for modeling biological processes such as neuronal response, and in mathematical finance.
- A stochastic process: $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$
- Discretized form: $x_{n+1} = x_n + \theta(\mu - x_n)\Delta t + \sigma \Delta W_n$
- W_t: Brownian motion, $\Delta W_n \sim N(0, \Delta t) = \sqrt{\Delta t} N(0, 1)$
- *Mean reverting process*, reverts exponentially at the rate θ.

Ornstein-Uhlenbeck (OU) Process

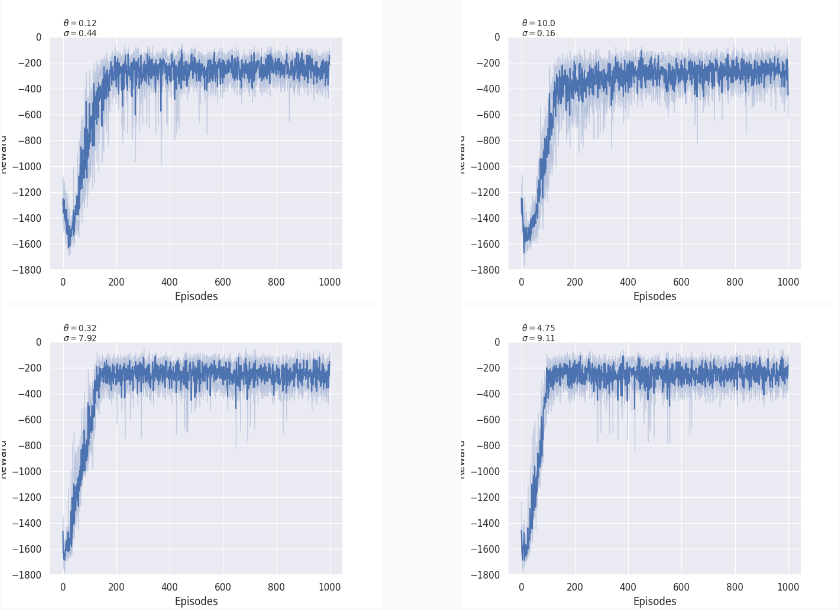
- Used for modeling biological processes such as neuronal response, and in mathematical finance.
- A stochastic process: $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$
- Discretized form: $x_{n+1} = x_n + \theta(\mu - x_n)\Delta t + \sigma \Delta W_n$
- W_t: Brownian motion, $\Delta W_n \sim N(0, \Delta t) = \sqrt{\Delta t} N(0, 1)$
- *Mean reverting process*, reverts exponentially at the rate θ.
- *Why* in DDPG? It's a diffusion type Markov process (and Normally distributed).

Parameters of OU process



- $\mu(= 0)$ is the long term mean.
- $\theta > 0$ is rate of mean reversion.
- $\sigma > 0$ is the volatility, per square-root t, of the random fluctuations.

DDPG on OpenAI Gym’s Pendulum



Soft Actor-Critic (SAC)[2]

- Off policy max entropy deep reinforcement learning with a stochastic actor.

[2] Haarnoja et al., 2018

Soft Actor-Critic (SAC)[2]

- Off policy max entropy deep reinforcement learning with a stochastic actor.
- Based on optimization of max entropy objective:

$$\pi^*(.|s_t) = argmax_\pi \mathbb{E}_\pi \left[\sum_t r(s_t, a_t) + \alpha H(\pi(.|s_t)) \right]$$

[2] Haarnoja et al., 2018

Soft Actor-Critic (SAC)[2]

- Off policy max entropy deep reinforcement learning with a stochastic actor.
- Based on optimization of max entropy objective:

$$\pi^*(.|s_t) = argmax_\pi \mathbb{E}_\pi \left[\sum_t r(s_t, a_t) + \alpha H(\pi(.|s_t)) \right]$$

- How to optimize it?

[2] Haarnoja et al., 2018

Optimizing the maximum entropy objective

- One solution: Q-learning but with entropy regularization. (Soft Q-learning)

Optimizing the maximum entropy objective

- One solution: Q-learning but with entropy regularization. (Soft Q-learning)
- It learns the soft Q-function directly but in continuous domains, optimal policies can be intractable.

Optimizing the maximum entropy objective

- One solution: Q-learning but with entropy regularization. (Soft Q-learning)
- It learns the soft Q-function directly but in continuous domains, optimal policies can be intractable.
- Better solution: Soft Actor-Critic. Learns a policy and it's Q-function combined.

Optimizing the maximum entropy objective

- One solution: Q-learning but with entropy regularization. (Soft Q-learning)
- It learns the soft Q-function directly but in continuous domains, optimal policies can be intractable.
- Better solution: Soft Actor-Critic. Learns a policy and it's Q-function combined.
- Similar to DDPG but with a stochastic actor.

Optimizing the maximum entropy objective

- One solution: Q-learning but with entropy regularization. (Soft Q-learning)
- It learns the soft Q-function directly but in continuous domains, optimal policies can be intractable.
- Better solution: Soft Actor-Critic. Learns a policy and it's Q-function combined.
- Similar to DDPG but with a stochastic actor.
- Sample efficient and stable.

Soft Actor-Critic: Policy Iteration

- Alternates between 2 steps (policy evaluation and improvement):

Soft Actor-Critic: Policy Iteration

- Alternates between 2 steps (policy evaluation and improvement):
  - Soft policy evaluation: Fix a policy and apply soft Bellman backup until it converges

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{s' \sim p_s, a' \sim \pi}\left[Q(s',a') - \log \pi(a'|s')\right]$$

Soft Actor-Critic: Policy Iteration

- Alternates between 2 steps (policy evaluation and improvement):
  - Soft policy evaluation: Fix a policy and apply soft Bellman backup until it converges

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{s' \sim p_s, a' \sim \pi}\left[Q(s',a') - \log \pi(a'|s')\right]$$

  - Soft policy improvement: Update policy through information projection

$$\pi_{new} = argmin_{\pi'} D_{KL}\left(\pi^{'}(.|s) || \frac{1}{Z} \exp Q^{\pi_{old}}(s,.)\right)$$

Soft Actor-Critic: Policy Iteration

- Alternates between 2 steps (policy evaluation and improvement):
    - Soft policy evaluation: Fix a policy and apply soft Bellman backup until it converges

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{s' \sim p_s, a' \sim \pi}\left[Q(s',a') - \log \pi(a'|s')\right]$$

    - Soft policy improvement: Update policy through information projection

$$\pi_{new} = argmin_{\pi'} D_{KL}\left(\pi^{'}(.|s) || \frac{1}{Z} \exp Q^{\pi_{old}}(s,.)\right)$$

- Can be shown that $Q^{\pi_{new}} \geq Q^{\pi_{old}}$

Soft Actor-Critic: Practical Implementation

- Soft actor-critic uses function approximators (neural nets) for actor (policy) and critic (Q function).

Soft Actor-Critic: Practical Implementation

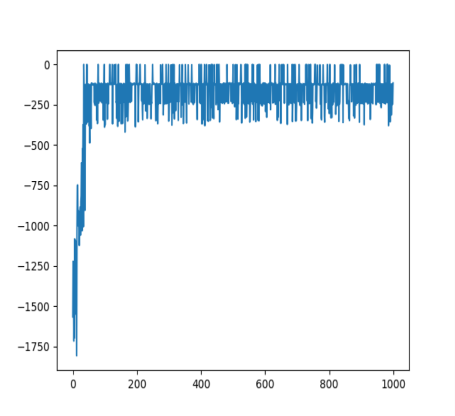
- Soft actor-critic uses function approximators (neural nets) for actor (policy) and critic (Q function).
- Optimizes the two steps (policy evaluation and improvement) together.

Soft Actor-Critic: Practical Implementation

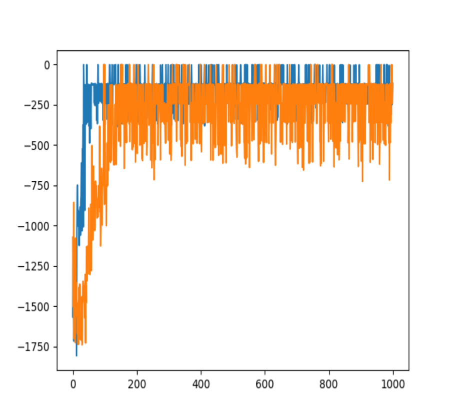
- Soft actor-critic uses function approximators (neural nets) for actor (policy) and critic (Q function).
- Optimizes the two steps (policy evaluation and improvement) together.
- Can compute unbiased stochastic gradients for both the steps using off policy data.

Results

SAC



SAC vs DDPG



Towards data efficient and stable RL: Conservative Value Iteration

Conservative Value Iteration (CVI)

- Optimizing KL divergence and entropy

$$W_{\tilde{\pi}}^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t\geq 0}^{\infty} \gamma^t \left(R_t - \tau \log \frac{\pi(A_t|S_t)}{\tilde{\pi}(A_t|S_t)} - \sigma \log \pi(A_t|S_t)\right)\middle| S_0 = s\right]$$

Conservative Value Iteration (CVI)

- Optimizing KL divergence and entropy

$$W_{\tilde{\pi}}^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t\geq 0}^{\infty}\gamma^{t}\left(R_t - \tau\log\frac{\pi(A_t|S_t)}{\widetilde{\pi}(A_t|S_t)} - \sigma\log\pi(A_t|S_t)\right)\middle| S_0 = s\right]$$

- Find policy that optimizes W

$$\pi^{\circ}(\cdot|s) = argmax_{\pi} W_{\tilde{\pi}}^{\pi}(s).$$

Conservative Value Iteration (CVI)

- Optimizing KL divergence and entropy

$$W^{\pi}_{\tilde{\pi}}(s) = \mathbb{E}^{\pi}\left[\sum_{t\geq 0}^{\infty} \gamma^t \left(R_t - \tau \log \frac{\pi(A_t|S_t)}{\widetilde{\pi}(A_t|S_t)} - \sigma \log \pi(A_t|S_t)\right) \middle| S_0 = s\right]$$

- Find policy that optimizes W

$$\pi^{\circ}(\cdot|s) = argmax_{\pi} W^{\pi}_{\tilde{\pi}}(s).$$

- Analogous to policy iteration.

Arriving at CVI

- CVI computes W_k defined by

$$W_k(s) = \mathbb{E}^{\pi_k}\left[\left(r + \gamma \mathbb{P} W_{k-1}\right)(s, A) - \tau \log \frac{\pi_k(A|s)}{\pi_{k-1}(A|s)} - \sigma \log \pi_k(A|s)\right],$$

Arriving at CVI

- CVI computes W_k defined by

$$W_k(s) = \mathbb{E}^{\pi_k}\left[\left(r + \gamma \mathbb{P} W_{k-1}\right)(s, A) - \tau \log \frac{\pi_k(A|s)}{\pi_{k-1}(A|s)} - \sigma \log \pi_k(A|s)\right],$$

- Find a policy that optimizes the W_k

Arriving at CVI

- CVI computes W_k defined by

$$W_k(s) = \mathbb{E}^{\pi_k}\left[\left(r + \gamma \mathbb{P} W_{k-1}\right)(s, A) - \tau \log \frac{\pi_k(A|s)}{\pi_{k-1}(A|s)} - \sigma \log \pi_k(A|s)\right],$$

- Find a policy that optimizes the W_k
- Analogous to value iteration.

- Let $\alpha = \tau/(\tau+\sigma)$ and $\beta := 1/(\tau+\sigma)$.[3]. π_k can be analytically obtained as

$$\pi_k(a|s) = \frac{\pi_{k-1}(a|s)^\alpha e^{\beta(r+\gamma\mathbb{P}W_{k-1})(s,a)}}{\sum_b \pi_{k-1}(b|s)^\alpha e^{\beta(r+\gamma\mathbb{P}W_{k-1})(s,b)}}.$$

[3] In soft actor-critic, $\tau = 0$, and thus, $\alpha = 0$ and $\beta = 1/\sigma = 1$.

- Let $\alpha = \tau/(\tau+\sigma)$ and $\beta := 1/(\tau+\sigma)$.[3]. π_k can be analytically obtained as

$$\pi_k(a|s) = \frac{\pi_{k-1}(a|s)^{\alpha} e^{\beta(r+\gamma\mathbb{P}W_{k-1})(s,a)}}{\sum_b \pi_{k-1}(b|s)^{\alpha} e^{\beta(r+\gamma\mathbb{P}W_{k-1})(s,b)}}.$$

- After some *dangerous* algebraic manipulations and defining action-value function as

$$\Psi_k(s,a) = (r+\gamma\mathbb{P}W_{k-1})(s,a) + \frac{\alpha}{\beta}\log\pi_{k-1}(a|s)$$

we get π_k as

$$\pi_k(a|s) = \frac{\exp(\beta\Psi_k(s,a))}{\sum_b \exp(\beta\Psi_k(s,b))}$$

and

$$W_k(s) = \mathbf{m}_\beta\Psi_k(s) = \mathbb{E}^{\pi_k}\left[\Psi_k(s,A) - \frac{1}{\beta}\log\pi_k(A|s)\right],$$

where $\mathbf{m}_\beta\Psi_k(s) := \beta^{-1}\log\sum_a \exp(\beta\Psi_k(s,a))$.

[3] In soft actor-critic, $\tau = 0$, and thus, $\alpha = 0$ and $\beta = 1/\sigma = 1$.

Arriving at CVI

The final (*not* so dangerous) update rules are:

$$\Psi_k(s,a) = r + \gamma \mathbb{P} W_{k-1}(s,a) + \alpha \left(\Psi_{k-1}(s,a) - W_{k-1}(s) \right)$$

$$\pi_k(a|s) = \frac{\exp\left(\beta \Psi_k(s,a)\right)}{\sum_b \exp\left(\beta \Psi_k(s,b)\right)}$$

$$W_k(s) = \mathbb{E}^{\pi_k}\left[\Psi_k(s,A) - \frac{1}{\beta} \log \pi_k(A|s) \right]$$

Conclusion

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.
- How to use chopsticks!

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.
- How to use chopsticks!
- Language and culture of Japan.

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.
- How to use chopsticks!
- Language and culture of Japan.
- Lunch and tea time discussions!

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.
- How to use chopsticks!
- Language and culture of Japan.
- Lunch and tea time discussions!
- Travel and hiking.

What did I learn?

- A *lot* of new RL/ML algorithms and recent advances in the field.
- ML tools: tensorflow, pytorch.
- A little neuroscience.
- How to use chopsticks!
- Language and culture of Japan.
- Lunch and tea time discussions!
- Travel and hiking.
- *List goes on...*

Did I forget something?

- Most importantly, how to do research!

Did I forget something?

- Most importantly, how to do research!
- Enjoyed a lot attending talks, seminars, meetings and listening to others work (mice experiments!)

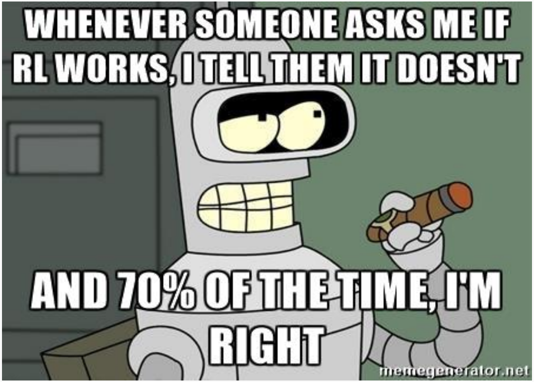
Did I forget something?

- Most importantly, how to do research!
- Enjoyed a lot attending talks, seminars, meetings and listening to others work (mice experiments!)
- and

Did I forget something?

- Most importantly, how to do research!
- Enjoyed a lot attending talks, seminars, meetings and listening to others work (mice experiments!)
- and
- 

Take Home



Thank You!

Acknowledgements

Arigato Gozaimasu!