

REINFORCEMENT LEARNING: THEORY AND PRACTICE

Modern Landscape: Part II

Profs. Amy Zhang and Peter Stone



TEXAS

The University of Texas at Austin

Logistics Questions?

Agenda for today

Natural progression of policy gradient and actor-critic deep RL methods:

Trust Region Policy Optimization

Proximal Policy Optimization

Deep Deterministic Policy Gradients

Twin Delayed DDPG

Maximum Entropy RL Framework

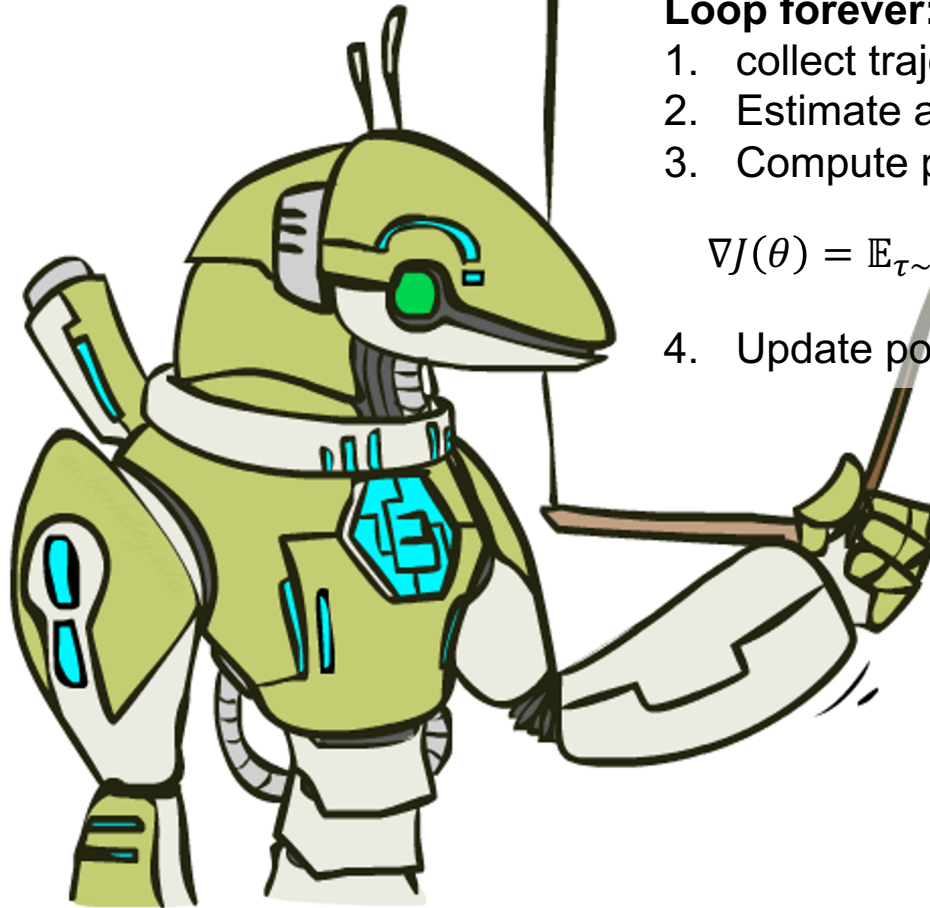
Soft Actor-Critic

Trust-Region Policy Optimization

Christopher Mutschler



Policy Gradients so far



Loop forever:

1. collect trajectories via policy π_θ
2. Estimate advantage function $A^{\pi_\theta}(a_t|s_t)$
3. Compute policy gradient:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(a_t|s_t) \right)$$

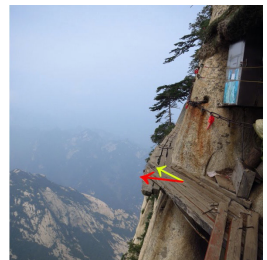
4. Update policy parameters $\theta_{new} \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

http://ai.berkeley.edu/lecture_slides.html

Policy Gradients so far

Problem

- Run gradient descent/ascent on one batch of collected experience
- Note: the advantage function (which is a noisy estimate) may not be accurate
 - Too large steps may lead to a disaster (even *if* the gradient is *correct*)
 - Too small steps are also bad
- Definition and scheduling of learning rates in RL is tricky as the underlying data distribution changes with updates to the policy
- Mathematical formulization:
 - First-order derivatives approximate the (parameter) surface to be flat
 - But if the surface exhibits high curvature it gets dangerous
 - **Projection: small changes in parameter space might lead to large changes in policy space!**
- Parameters θ get updated to areas too far out of the range from where previous data was collected (*note: a bad policy leads to bad data*)



Images taken from https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04e00000 and <http://www.taiwanoffthebeatentrack.com/2012/08/23/mount-hua-华山-the-most-dangerous-hike-in-the-world/>

Trust-Region Policy Optimization (TRPO)

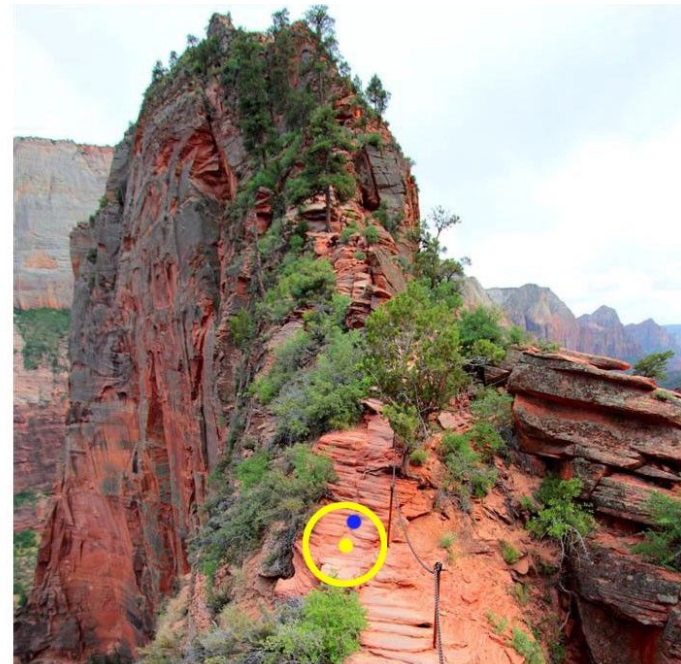
“Simple” Idea

Regularize updates to the policy parameters,
such that the policy does not change too much.

Motivation: Why trust region optimization?



Line search
(like gradient ascent)



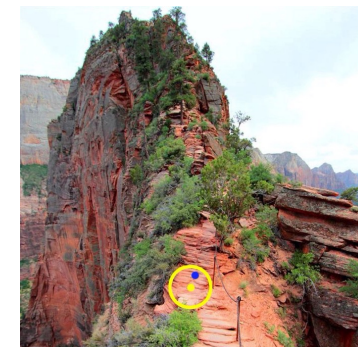
Trust region

Primer: Trust-Region Methods

Optimization in Machine Learning: two classes

1. Line Search, e.g., gradient descent
 - find a (some) direction of improvement
 - (cleverly) select a step length

2. Trust-Region Methods
 - select a trust region (analog to max step length)
 - find a point of improvement in that region



Primer: Trust-Region Methods

- **Idea:**

- Approximate the real objective f with something simpler, i.e., \tilde{f}
- Solve $\tilde{x}^* = \arg \min_x \tilde{f}(x)$

- **Problem:**

- The optimum \tilde{x}^* might be in a region where \tilde{f} poorly approximates f
- \tilde{x}^* might be far from optimal

- **Solution:**

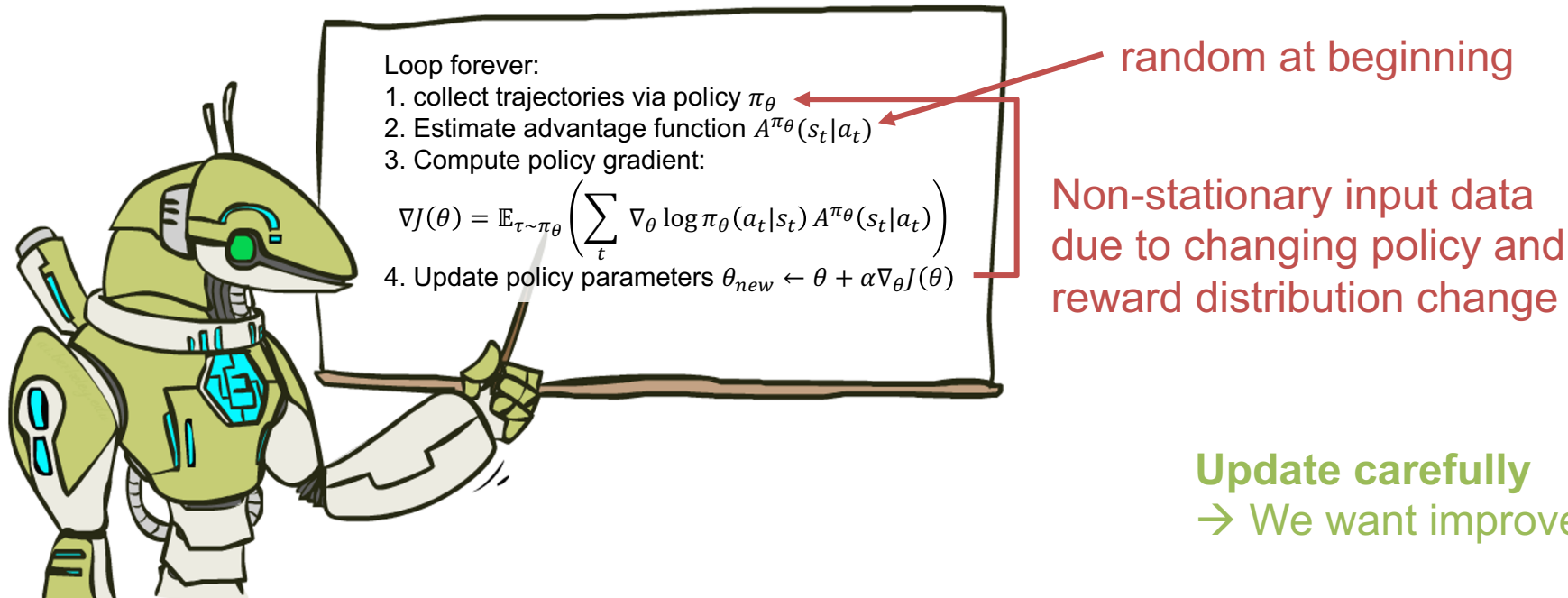
- Restrict the search to a region tr where we trust \tilde{f} to approximate f well
- Solve $\tilde{x}^* = \arg \min_{x \in tr} \tilde{f}(x)$

Trust-Region Policy Optimization (TRPO)

So back to what we actually do...

The problem(s) of the Policy Gradient (PG) is that

- PG keeps old and new policy close in parameter space, while
- small changes can lead to large differences in performance, and
- “large” step-sizes hurt performance (whatever “large” means...)



Loop forever:

1. collect trajectories via policy π_θ
2. Estimate advantage function $A^{\pi_\theta}(s_t|a_t)$
3. Compute policy gradient:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t|a_t) \right)$$
4. Update policy parameters $\theta_{new} \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

random at beginning

Non-stationary input data due to changing policy and reward distribution change

Update carefully
 → We want improvement and not degradation

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π :

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot | s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy¹:

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right]$$

Expected return of the new policy

Expected return of the old policy

Sample from new policy

¹ Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π :

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy¹:

$$\begin{aligned} \eta(\pi_{new}) &= \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right] \\ &= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s, a) \end{aligned}$$

Discounted visitation frequency according to new policy:

$$\rho_{\pi_{new}}(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \dots$$

¹ Schulman et al.: Trust-Region Policy Optimization. ICML 2015.

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π :

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy:

$$\begin{aligned} \eta(\pi_{new}) &= \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right] \\ &= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \underbrace{\pi_{new}(a|s) A_{\pi_{old}}(s, a)}_{> 0} \end{aligned}$$

↑ new expected return > ↑ old expected return

If we can guarantee this...

→ New objective guarantees improvement from $\pi_{old} \rightarrow \pi_{new}$

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π :

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy:

$$\begin{aligned} \eta(\pi_{new}) &= \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right] \\ &= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s, a) \end{aligned}$$

However, this cannot be easily estimated. The state visitations that we sampled so far are coming from the old policy!

→ we cannot optimize this in the current form!

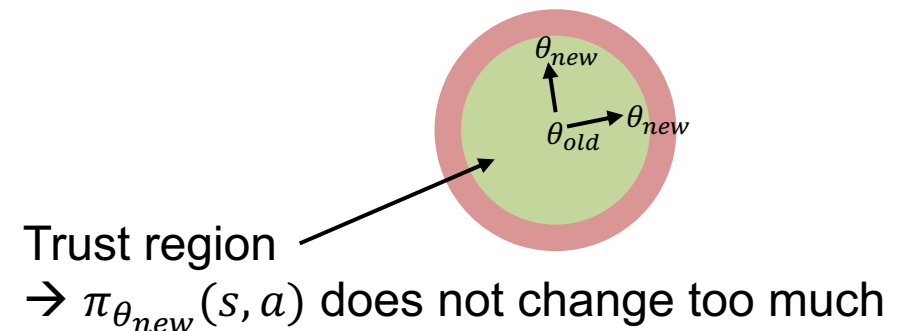
Trust-Region Policy Optimization (TRPO)

$$\begin{aligned}
 \eta(\pi_{new}) &= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s, a) \\
 &\approx L(\pi_{new}) = \eta(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s, a)
 \end{aligned}$$

approximate locally

This we already sampled → We already have this!

- The approximation is accurate within step size δ (trust region)
 - δ needs to be chosen based on a lower-bound approximation error
- Monotonic improvement guaranteed
 - (within the green region!)



Trust-Region Policy Optimization (TRPO)

- If we want to optimize $L(\theta_{new})$ instead of $\eta(\theta_{new})$...
with a guarantee of monotonic improvement on $\eta(\theta_{new})$, ...
... we need a bound on $L(\theta_{new})$.
- It can be proven that there exists the following bound^{1,2}:

$$\eta(\pi_{new}) \geq L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new}), \text{ where } C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$$

¹ Schulman et al.: Trust-Region Policy Optimization. ICML 2015.

² Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.

Trust-Region Policy Optimization (TRPO)

- A monotonically increasing policy can be defined by (minorization-maximization algorithm):

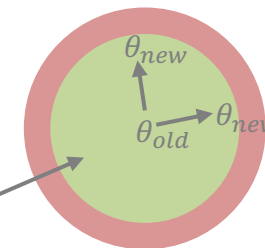
$$\pi = \arg \max_{\pi} [L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new})], \text{ where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$$

Side-note:

- A constraint on the KL-divergence between new and old policy (i.e., a trust region constraint) allows larger step sizes while being mathematically equivalent:

$$\pi = \arg \max_{\pi} L_{\pi_{old}}, \text{ such that } D_{KL}^{max}(\pi_{old}, \pi) \leq \delta$$

- Approximation with L is accurate within δ
 → here, monotonic improvement guaranteed



Trust region
 → $\pi_{\theta_{new}}(s, a)$ does not change too much

Limitations of TRPO

- Hard to use with architectures with multiple outputs, e.g., policy and value function (need to weight different terms in distance metric)
- Empirically performs poorly on tasks requiring deep CNNs and RNNs, e.g., Atari benchmark (more suitable for locomotion)
- Conjugate gradients makes implementation more complicated than SGD

Reading Questions: TRPO

Michelle Ding

[Trust Region Policy Optimization] What is achieved by keeping old and new policies sufficiently close together? Why is this not guaranteed by the policy gradient theorem from Chapter 13, which I thought ensures the convergence of normal policy gradient? i.e. If a single bad step can collapse performance, how is convergence guaranteed?

Alperen Duru

Didn't we already discuss in the policy gradient methods that one such advantage of using the policy gradient methods is to smooth out the performance changes compared to the action-value methods? Why does TRPO discuss the normal policy gradient methods that a single small step can collapse the policy performance?

PPO (clipping version)

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}(s, a)}, g(\epsilon, A^{\pi_{\theta_k}(s, a)}) \right),$$

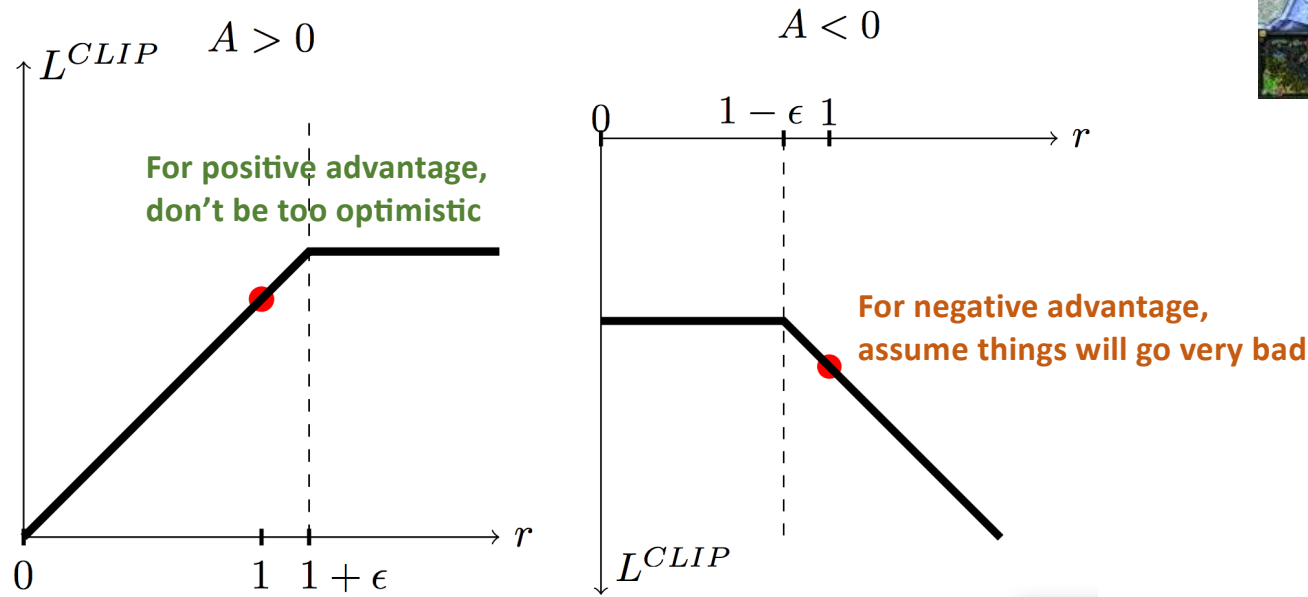
where

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases}$$

Proximal Policy Optimization (PPO)



Dota 2



$$r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right]$$

Reading Questions

Young-ho

Compared to TRPO, does PPO converge faster than TRPO?

-PPO: first-order

-TRPO: complex second-order

I understand that speed can be faster than TRPO, but how can performance be better than TRPO, even with a less complex model?

Laith Altarabishi

(PPO) Suppose that the advantage is negative and the new policy value increases massively in value - ie the action becomes a lot more likely. Since the clipping is a maximization term - doesn't this mean that the objective would see a massive update that is not controlled? Are we only concerned with the case where the objective increases - if so why?

Reading Questions

Saloni Modi

Does the epsilon in PPO have to be constant? I feel like I'd be cool with the new policy initially getting really far from the original policy but then overtime decaying the acceptable distance.

Deterministic Policy Gradient (DPG)

Objective

$$J(\mu_\theta) = \mathbb{E}_{s \sim \rho^\mu} [Q^\mu(s, \mu_\theta(s))]$$

$\mu_\theta(s)$ Deterministic policy function (e.g., if action is continuous)

$Q^\mu(s, a)$ Q-function following policy μ

Deterministic Policy Gradient (DPG)

Taking Derivative w.r.t θ :

$$\nabla_{\theta} J(\mu_{\theta}) = \mathbb{E}_{s \sim \rho^{\mu}} \left[\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu}(s, a) \Big|_{a=\mu(s)} \right]$$

No need to take gradient w.r.t μ
Because of deterministic policy
gradient theorem

Scalar gradient at (s, a)

Sample state from the distribution, $s_i \sim \rho^{\mu}$:

$$\nabla_{\theta} J(\mu_{\theta}) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \mu_{\theta}(s_i) \nabla_a Q^{\mu}(s_i, a) \Big|_{a=\mu(s_i)}$$

DDPG (Deep Deterministic Policy Gradient)

- Use deep networks to represent policy / Q.
- Generate trajectories with current policy + noise
 - Since the policy is deterministic
- Save trajectories into replay buffer and sample from it (Off-policy!)
- Learn Q^μ via DQN using target network
- Learn μ using the slide above.

Reading Questions

Justin Sasek

(DDPG) Why is importance sampling not needed even though DDPG is off-policy?

Distributed Distributional Deterministic Policy Gradients (D4PG)

- Distributional version of DDPG
 1. Distributional critic
 2. N-step returns
 3. Multiple distributed parallel actors
 4. Prioritized experience replay

Twin Delayed DDPG (TD3)

Clipped Double Q-learning (CDQ)

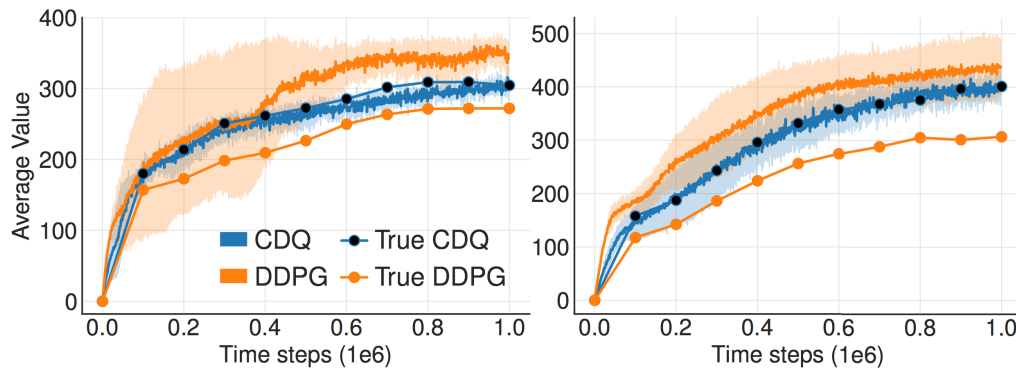
$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \underbrace{\pi_{\phi_1}(s')}_{\text{Policy}})$$

Two independent models θ_1, ψ_1 and θ_2, ψ_2

Policy and value updates too tightly coupled

Delayed update of Target and Policy Networks

Target Policy Smoothing



(a) Hopper-v1

(b) Walker2d-v1

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

Stochastic Policies



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: $R(s, a) + \lambda H(\pi(\cdot | s))$
- Transition: $\Pr(s' | s, a)$
- Discount: γ



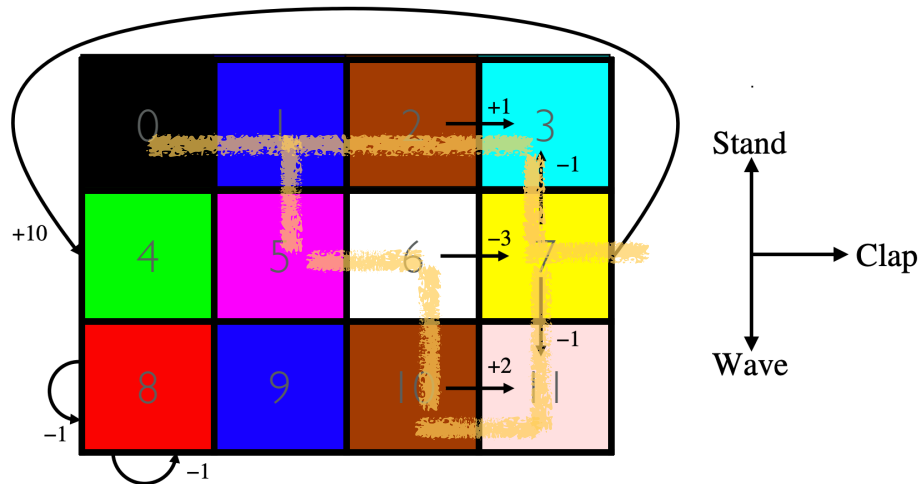
Exercise: Max Ent RL

Reward: $R(s, a) + \lambda H(\pi(\cdot | s))$

$$H(X) := - \sum_{x \in \mathcal{X}} p(x) \log p(x).$$

How do the rewards change?

$\lambda = 1$



	Stand	Clap	Wave
0	-1	0	0
1	-1	0	0
2	-1	1	0
3	-1	-1	0
4	0	0	0
5	0	0	0
6	0	-3	0
7	-1	10	-1
8	0	0	-1
9	0	0	-1
10	0	2	-1
11	0	0	-1

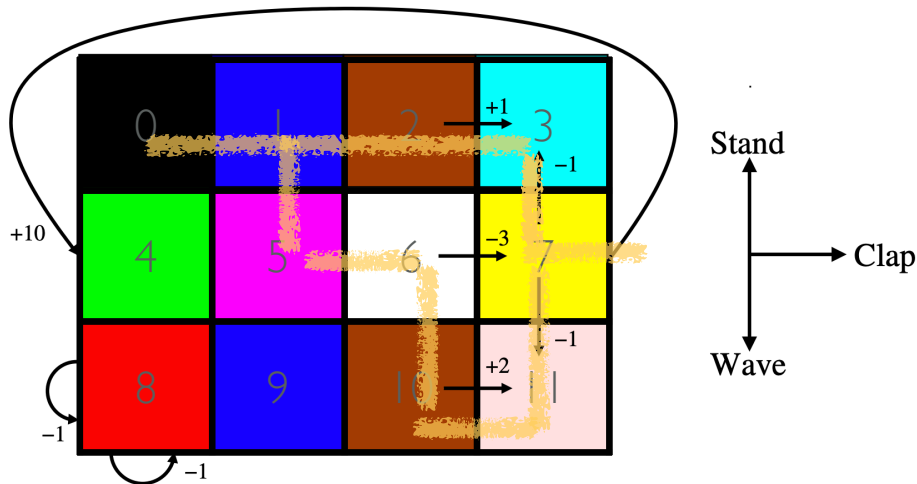
Exercise: Max Ent RL

Reward: $R(s, a) + \lambda H(\pi(\cdot | s))$

$$H(X) := - \sum_{x \in \mathcal{X}} p(x) \log p(x).$$

How do the rewards change?

$\lambda = 1$



	Stand	Clap	Wave
0	-1	0	0
1	$-1 - \log(0.5)$	$-\log(0.5)$	$-\log(0.5)$
2	-1	1	0
3	-1	-1	0
4	0	0	0
5	$-\log(0.5)$	$-\log(0.5)$	$-\log(0.5)$
6	0	-3	0
7	$-1 - \log(1/3)$	$10 - \log(1/3)$	$-1 - \log(1/3)$
8	0	0	-1
9	0	0	-1
10	$-\log(0.5)$	$2 - \log(0.5)$	$-1 - \log(0.5)$
11	0	0	-1

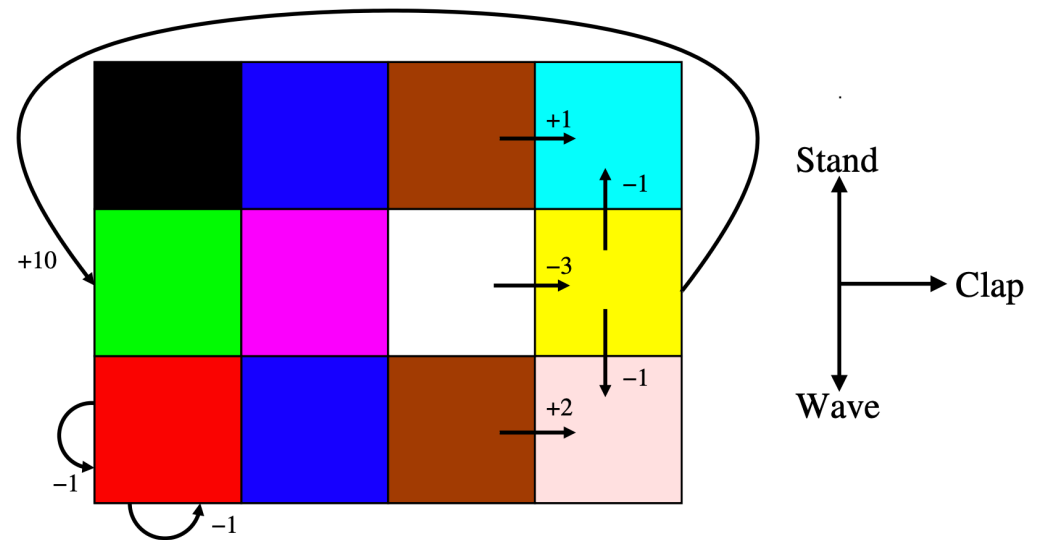
Is this a valid reward function?

Is this a valid reward function?

No! It is non stationary because it depends on the current policy.

Exercise: Max Ent RL

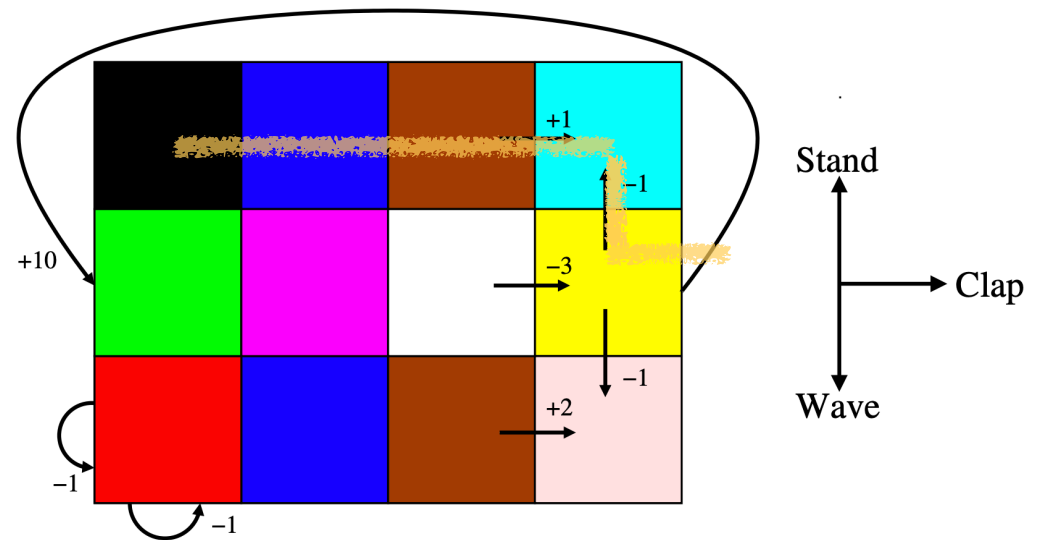
What does the optimal policy look like that *also maximizes entropy* if $\gamma = 0.9$? Assume episodic setting where episode ends when +10 is reached.



Exercise: Max Ent RL

What does the optimal policy look like that *also maximizes entropy* if $\gamma = 0.9$? Assume episodic setting where episode ends when +10 is reached.

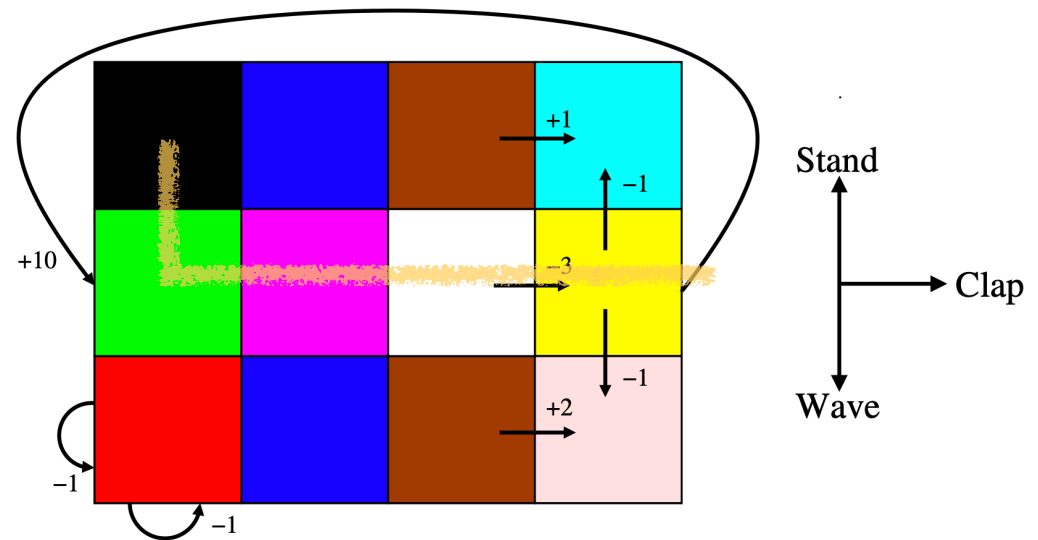
$$V(S = 0) = 0 + 0.9 * 0 + 0.9^2 * 1 + 0.9^3 * 0 + 0.9^4 * 10 = 7.371$$



Exercise: Max Ent RL

What does the optimal policy look like that *also maximizes entropy* if $\gamma = 0.9$? Assume episodic setting where episode ends when +10 is reached.

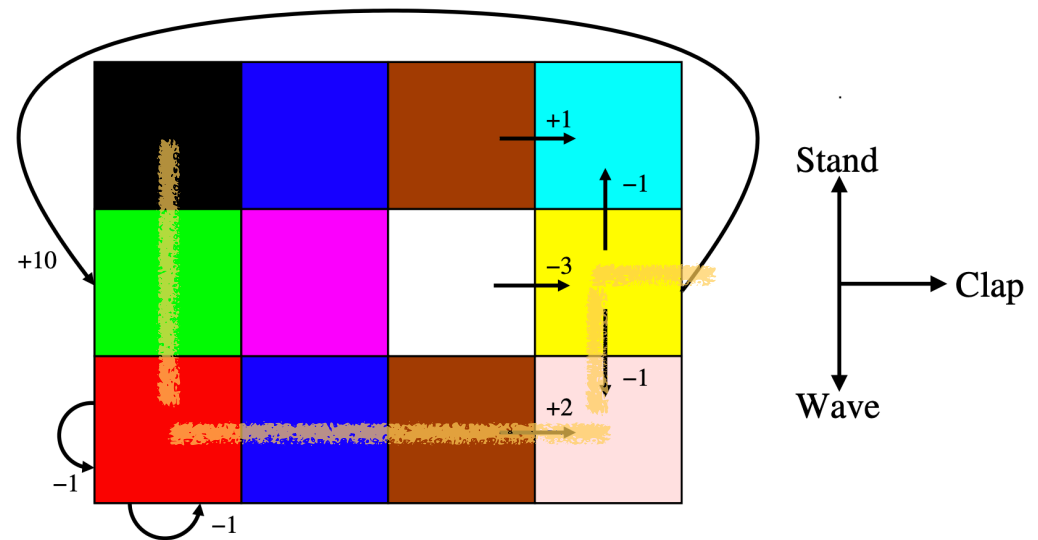
$$V(S = 0) = 0 + 0.9 * 0 + 0.9^2 * 1 - 0.9^3 * 3 + 0.9^4 * 10 = 5.184$$



Exercise: Max Ent RL

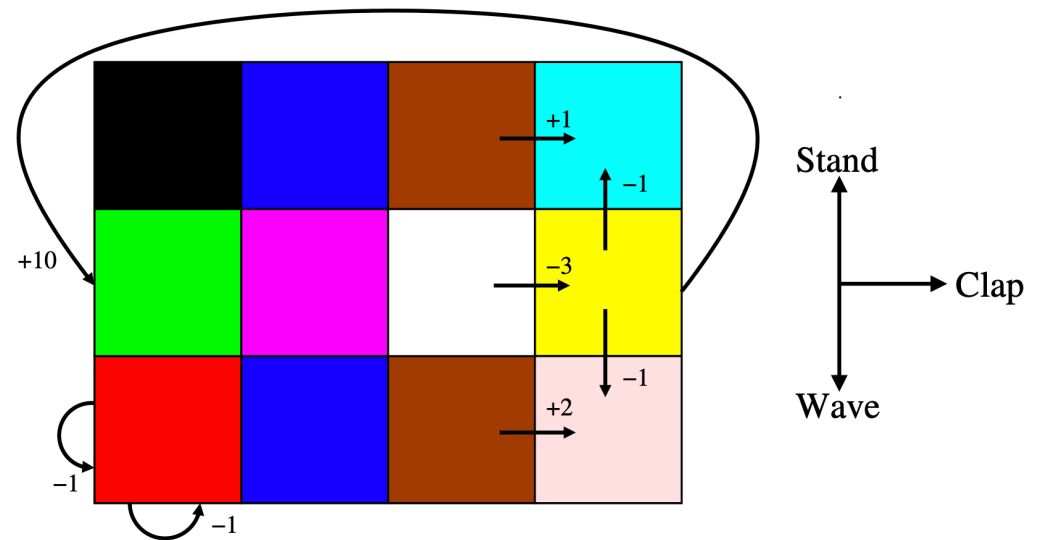
What does the optimal policy look like that *also maximizes entropy* if $\gamma = 0.9$? Assume episodic setting where episode ends when +10 is reached.

$$V(S = 0) = 0 + 0.9 * 0 + 0.9^2 * 1 - 0.9^3 * 3 + 0.9^4 * 2 + 0.9^5 * 0 + 0.9^6 * 10 = 5.25$$



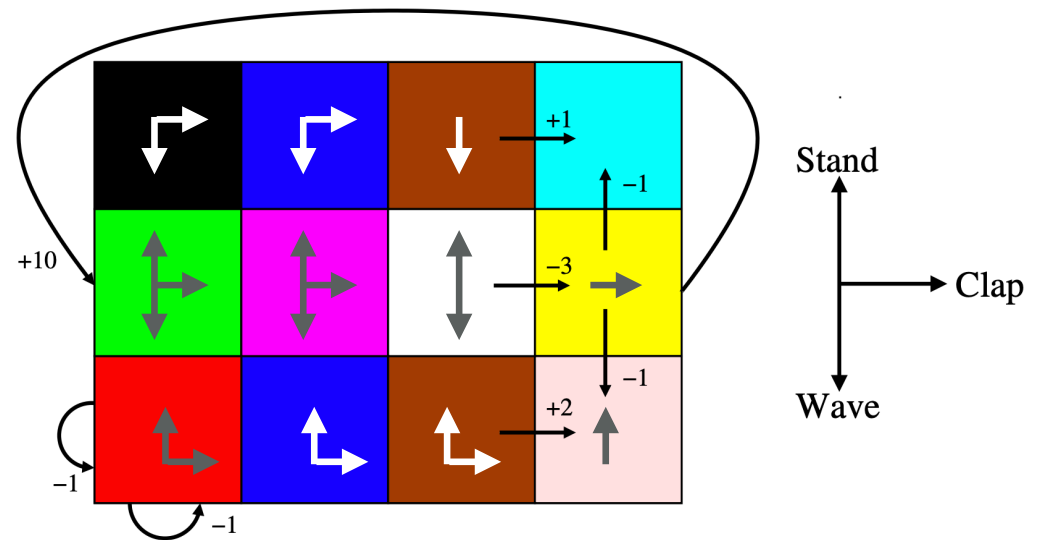
Exercise: Max Ent RL

What does the optimal policy look like that *also maximizes entropy* if $\gamma = 1$?



Exercise: Max Ent RL

What does the optimal policy look like that *also maximizes entropy* if $\gamma = 1$?

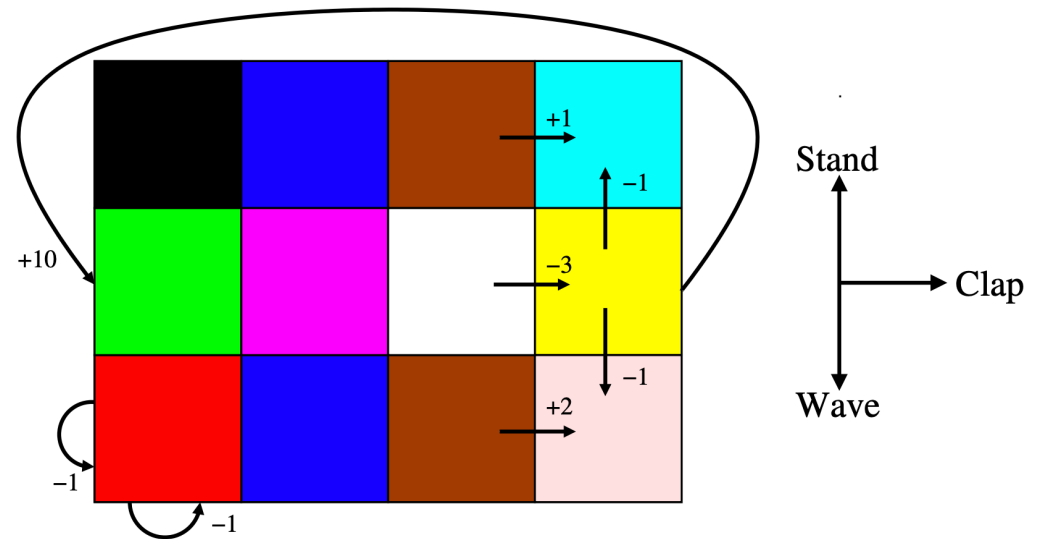


Exercise: Max Ent RL

Is that the same optimal policy for this problem?

$\gamma = 1$

Reward: $R(s, a) + \lambda H(\pi(\cdot | s))$



Exercise: Max Ent RL

Is that the same optimal policy for this problem?

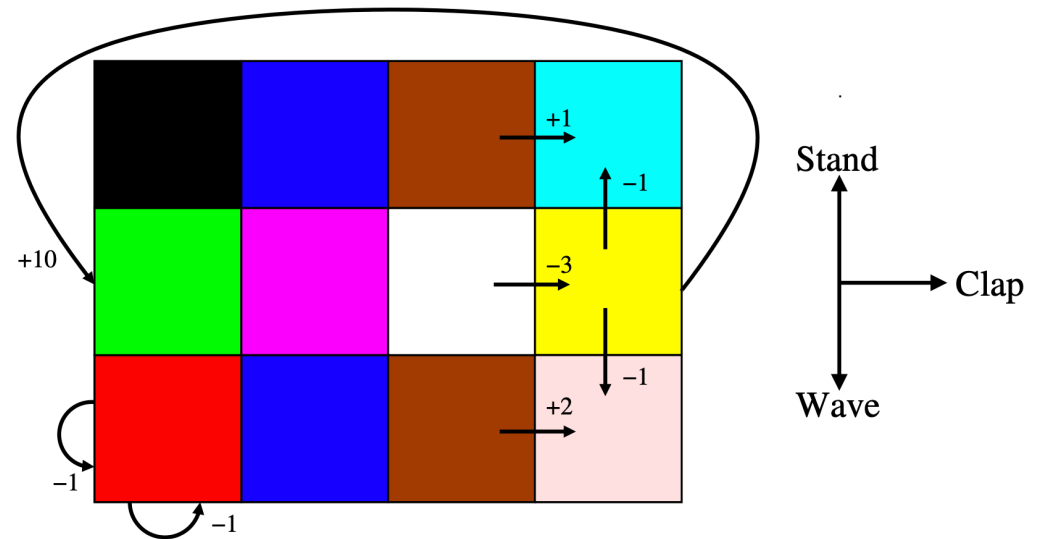
$\gamma = 1$



Lagrangian dual form of previous objective

Reward: $R(s, a) + \lambda H(\pi(\cdot | s))$

Not necessarily, and it depends on λ



Optimal Policy

- Standard MDP

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

- Soft MDP

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



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

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



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



Greedy Policy

- Standard MDP (deterministic policy)

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

- Soft MDP (stochastic policy)

$$\begin{aligned}\pi_{greedy}(\cdot | s) &= \operatorname{argmax}_{\pi} \sum_a \pi(a|s) Q(s, a) + \lambda H(\pi(\cdot | s)) \\ &= \frac{\exp(Q(s, \cdot) / \lambda)}{\sum_a \exp(Q(s, a) / \lambda)} = \operatorname{softmax}(Q(s, \cdot) / \lambda)\end{aligned}$$

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

$$Q_{soft}^{\pi_i}(s, a) \leftarrow R(s, a)$$

$$+ \gamma \sum_{s'} \Pr(s'|s, a) \left[\sum_{a'} \pi_i(a'|s') Q_{soft}^{\pi_i}(s', a') + \lambda H(\pi_i(\cdot | s')) \right] \quad \forall s, a$$

Policy improvement:

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

$i \leftarrow i + 1$

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



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 \mathbf{w} , $\bar{\mathbf{w}}$, 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 $\hat{a}' \sim \pi_\theta(\cdot | \hat{s}')$

$$\text{Gradient: } \frac{\partial \text{Err}}{\partial \mathbf{w}} = \left[Q_{\mathbf{w}}^{\text{soft}}(\hat{s}, \hat{a}) - \hat{r} - \gamma [Q_{\bar{\mathbf{w}}}^{\text{soft}}(\hat{s}', \hat{a}') + \lambda H(\pi_\theta(\cdot | \hat{s}'))] \right] \frac{\partial Q_{\mathbf{w}}^{\text{soft}}(\hat{s}, \hat{a})}{\partial \mathbf{w}}$$

$$\text{Update weights: } \mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \text{Err}}{\partial \mathbf{w}}$$

$$\text{Update policy: } \theta \leftarrow \theta - \alpha \frac{\partial \text{KL}(\pi_\theta | \text{softmax}(Q_{\bar{\mathbf{w}}}^{\text{soft}} / \lambda))}{\partial \theta}$$

Update state: $s \leftarrow s'$

Every c steps, update target: $\bar{\mathbf{w}} \leftarrow \mathbf{w}$



Final Logistics

Next lecture:

Abstractions and Options

Reading assignments due **2PM Monday**

Final project literature review due at **11:59pm on Thursday, 4/11**

Complete Programming Assignment for Chapters 12+13 on edx by Sunday 11:59 PM CST