CS4400

DEEP REINFORCEMENT LEARNING

Lecture 7: Off-Policy Actor-Critic

Wendelin Böhmer

<j.w.bohmer@tudelft.nl>

TU Delft

12th of December 2023

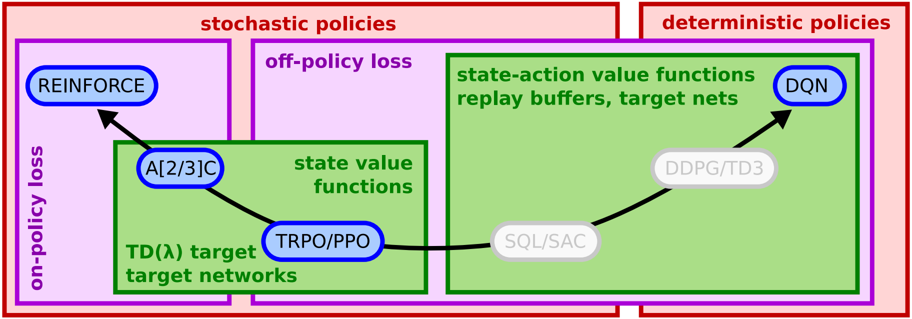
Content of this lecture

7.1

Off-Policy Actor-Critic
Deterministic policies

7.1 Deep RL algorithms so far

- Spectrum from REINFORCE to DQN
  - Algorithm choice depends how well value functions generalize
- Empirically PPO best actor-critic algorithms so far
  - allows off-policy training on batches of on-policy samples



core concept: Deterministic policy gradients

- Continuous actions $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$

$$\nabla_\theta \mathcal{L}_\pi[\theta] \quad \approx \quad -\nabla_\theta \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1} \gamma^t \Big(\overbrace{Q^\pi(s_t, \boldsymbol{a}_t) - V^\pi(s_t)}^{A^\pi(s_t, \boldsymbol{a}_t)} \Big) \frac{\pi_\theta(\boldsymbol{a}_t|s_t)}{\mu(\boldsymbol{a}_t|s_t)}\Big]$$

approximation drops $\frac{\xi_t^\pi(s_t)}{\xi^\mu(s_t)}$, see Sutton et al. (2016) for ways to estimate it (DPG, Silver et al., 2014)

core concept: Deterministic policy gradients

- Continuous actions $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$

$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\pi[\theta] \quad &\approx \quad -\nabla_\theta \mathbb{E}_\mu \Big[\sum_{t=0}^{n-1} \gamma^t \big(\overbrace{Q^\pi(s_t, \boldsymbol{a}_t) - V^\pi(s_t)}^{A^\pi(s_t, \boldsymbol{a}_t)} \big) \frac{\pi_\theta(\boldsymbol{a}_t | s_t)}{\mu(\boldsymbol{a}_t | s_t)} \Big] \\
&= \quad -\nabla_\theta \sum_{t=0}^{n-1} \iint \xi_t^\mu(s_t)\, \pi_\theta(\boldsymbol{a}_t | s_t)\, \gamma^t Q^\pi(s_t, \boldsymbol{a}_t)\, ds_t\, d\boldsymbol{a}_t
\end{aligned}
$$

approximation drops $\frac{\xi_t^\pi(s_t)}{\xi^\mu(s_t)}$, see Sutton et al. (2016) for ways to estimate it

(DPG, Silver et al., 2014)

core concept: Deterministic policy gradients

- Continuous actions $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$
- Deterministic policy $\boldsymbol{\pi}_\theta : \mathcal{S} \to \mathcal{A}$
  - equivalent to $\pi_\theta\big(\boldsymbol{\pi}_\theta(s)|s\big) = 1, \forall s \in \mathcal{S}$

$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\pi[\theta] \quad &\approx \quad -\nabla_\theta \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1} \gamma^t \Big(\overbrace{Q^\pi(s_t, \boldsymbol{a}_t) - V^\pi(s_t)}^{A^\pi(s_t,\boldsymbol{a}_t)}\Big) \tfrac{\pi_\theta(\boldsymbol{a}_t|s_t)}{\mu(\boldsymbol{a}_t|s_t)}\Big] \\
&= \quad -\nabla_\theta \sum_{t=0}^{n-1} \iint \xi_t^\mu(s_t)\, \pi_\theta(\boldsymbol{a}_t|s_t)\, \gamma^t Q^\pi(s_t, \boldsymbol{a}_t)\, ds_t\, d\boldsymbol{a}_t \\
&= \quad -\sum_{t=0}^{n-1} \int \xi_t^\mu(s_t)\, \gamma^t\, \nabla_\theta Q^\pi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\, ds_t
\end{aligned}
$$

approximation drops $\frac{\xi_t^\pi(s_t)}{\xi^\mu(s_t)}$, see Sutton et al. (2016) for ways to estimate it (DPG, Silver et al., 2014)

core concept: Deterministic policy gradients

- Continuous actions $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$
- Deterministic policy $\boldsymbol{\pi}_\theta : \mathcal{S} \to \mathcal{A}$
  - equivalent to $\pi_\theta\big(\boldsymbol{\pi}_\theta(s)|s\big) = 1, \forall s \in \mathcal{S}$

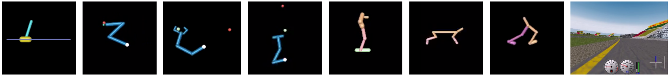
$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\pi[\theta] \quad &\approx \quad -\nabla_\theta \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1} \gamma^t \big(\overbrace{Q^\pi(s_t, \boldsymbol{a}_t) - V^\pi(s_t)}^{A^\pi(s_t, \boldsymbol{a}_t)}\big) \frac{\pi_\theta(\boldsymbol{a}_t|s_t)}{\mu(\boldsymbol{a}_t|s_t)}\Big] \\
&= \quad -\nabla_\theta \sum_{t=0}^{n-1} \iint \xi_t^\mu(s_t)\, \pi_\theta(\boldsymbol{a}_t|s_t)\, \gamma^t Q^\pi(s_t, \boldsymbol{a}_t)\, ds_t\, d\boldsymbol{a}_t \\
&= \quad -\sum_{t=0}^{n-1} \int \xi_t^\mu(s_t)\, \gamma^t\, \nabla_\theta Q^\pi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\, ds_t \\
&\propto \quad -\nabla_\theta \mathbb{E}_\mu\Big[Q^\pi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\Big|\, t \sim p(\cdot|\gamma)\Big], \qquad p(t|\gamma) \propto \gamma^t
\end{aligned}
$$

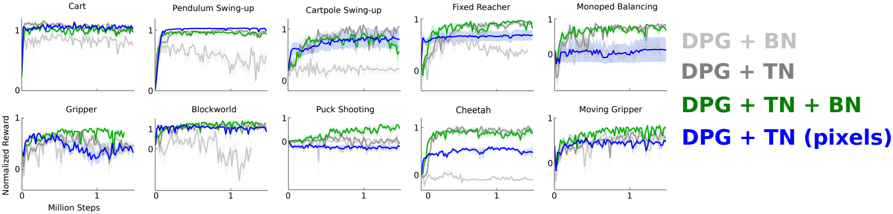
- Implementations based on transitions often omit γ^t

$p(t|\gamma)$ is technically a `Conway-Maxwell-Poison distribution` with $\lambda = \gamma$ and $\nu = 0$. (DPG, Silver et al., 2014)

Deep DPG

- Off-policy DPG with:
  - Experience replay buffer
  - Actor and critic target networks (TN)
  - Soft target update
  - Batch-norm over inputs (BN)
  - Exploration by adding noise to $\pi_\theta(s_t)$





spinningup.openai.com/en/latest/algorithms/ddpg.html

DDPG and images/results by Lillicrap et al. (2016)

Question: Implementing DPG

- For actor network $\boldsymbol{\pi}_\theta(s)$ and critic network $Q_\phi(s, \boldsymbol{a})$:

$$\nabla_\theta \mathcal{L}_\pi[\theta] \quad \approx \quad -\sum_{t=0}^{n-1} \gamma^t \, \mathbb{E}_\mu\Big[\nabla_\theta Q_\phi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\Big]$$

- How can we compute the gradients for θ and ϕ in PyTorch?

Question: Implementing DPG

- For actor network $\boldsymbol{\pi}_\theta(s)$ and critic network $Q_\phi(s, \boldsymbol{a})$:

$$\nabla_\theta \mathcal{L}_\pi[\theta] \quad \approx \quad -\sum_{t=0}^{n-1} \gamma^t \, \mathbb{E}_\mu\Big[\nabla_\theta Q_\phi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\Big]$$

- How can we compute the gradients for θ and ϕ in PyTorch?
- each network needs its own optimizer

$$\arg\min_\phi \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1}\Big(r_t + \gamma\, Q_{\phi'}\big(s_{t+1}, \boldsymbol{\pi}_{\theta'}(s_{t+1})\big) - Q_\phi(s_t, a_t)\Big)^2\Big]$$

$$\arg\min_\theta -\mathbb{E}_\mu\Big[\sum_{t=0}^{n-1}\gamma^t Q_\phi\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\Big]$$

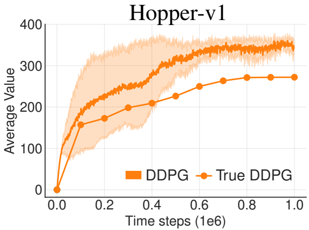
- Try to write the above gradient descend problems in PyTorch!
  - given a mini-batch `batch = self.replay_buffer.sample()`
  - given an `actor(batch['states'])`
  - given a `critic(batch['states'], batch['actions'])`

7.1 DDPG implementation

```
from torch.optim import Adam
from torch.nn.functional import mse_loss
actor_optimizer = Adam(actor.parameters())
critic_optimizer = Adam(critic.parameters())
actor_t, critic_t = deepcopy(actor), deepcopy(critic)
for _ in range(max_updates):
    batch = self.replay_buffer.sample()
    # Critic update
    q_values = critic(batch['states'], batch['actions'])
    next_actions = actor_t(batch['next_states'])
    targets = batch['rewards'] + gamma * (~batch['terminals'] \
                * critic_t(batch['next_states'], next_actions))
    critic_optimizer.zero_grad()
    mse_loss(q_values, targets.detach()).backward()
    critic_optimizer.step()
    # Actor update
    q_values = critic(batch['states'], actor(batch['states']))
    actor_optimizer.zero_grad()
    (-q_values * batch['discounts']).mean().backward()
    actor_optimizer.step()
    # Target network updates
    actor_t = target_model_updates(actor_t, actor)
    critic_t = target_model_updates(critic_t, critic)
```

7.1 Twin delayed DDPG (TD3)

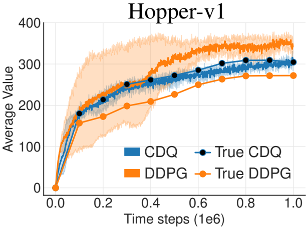
- $\mathbb{E}\big[\max_\theta Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big] \geq \max_\theta \mathbb{E}\big[Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big]$
  - like in Double Q-learning (Lecture 5)



images and results from Fujimoto et al. (TD3, 2018)

7.1 Twin delayed DDPG (TD3)

- $\mathbb{E}\big[\max_\theta Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big] \geq \max_\theta \mathbb{E}\big[Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big]$
  - like in Double Q-learning (Lecture 5)
- CDQ: two critics against overestimation
  - minimum, *unlike* in Double Q-learning!

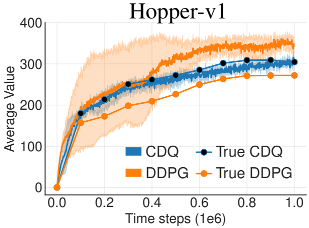


$$\mathcal{L}_i^Q[\phi_i] \quad = \quad \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1}\Big(r_t + \gamma \min_{j\in\{1,2\}} Q_{\phi'_j}\big(s_{t+1}, \boldsymbol{\pi}_{\theta'}(s_{t+1})\big) - Q_{\phi_i}(s_t, a_t)\Big)^2\Big]$$

Clipped Double Q-learning (CDQ) is Q-learning with two Q-value functions   images and results from Fujimoto et al. (TD3, 2018)

7.1 Twin delayed DDPG (TD3)

- $\mathbb{E}\big[\max_\theta Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big] \geq \max_\theta \mathbb{E}\big[Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big]$
  - like in Double Q-learning (Lecture 5)
- CDQ: two critics against overestimation
  - minimum, *unlike* in Double Q-learning!
- TD3: regularization with clipped noise
  $\boldsymbol{\epsilon} \sim \mathrm{clip}\big(\mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2), -\boldsymbol{c}, \boldsymbol{c}\big) \in \mathbb{R}^{|\mathcal{A}|}$



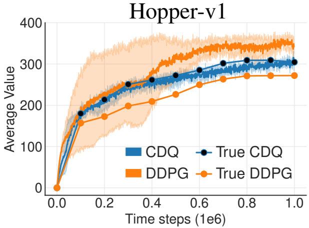
$$\mathcal{L}_i^Q[\phi_i] \quad = \quad \mathbb{E}_\mu\Big[\sum_{t=0}^{n-1}\Big(r_t + \gamma \min_{j\in\{1,2\}} Q_{\phi'_j}\big(s_{t+1}, \boldsymbol{\pi}_{\theta'}(s_{t+1}) + \boldsymbol{\epsilon}\big) - Q_{\phi_i}(s_t, a_t)\Big)^2\Big]$$

- TD3 algorithm one of the highest performing DDPG variants
  - introduces pessimistic values ($\rightarrow$ Lecture 9)

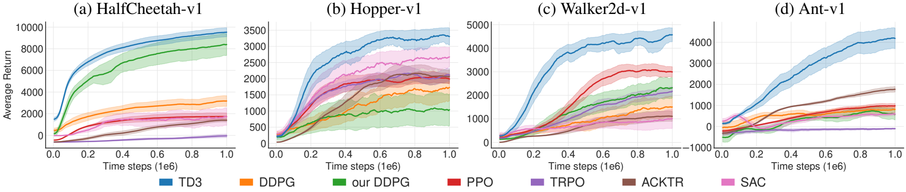
spinningup.openai.com/en/latest/algorithms/td3.html images and results from Fujimoto et al. (TD3, 2018)

7.1 Twin delayed DDPG (TD3)

- $\mathbb{E}\big[\max_\theta Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big] \geq \max_\theta \mathbb{E}\big[Q\big(s, \boldsymbol{\pi}_\theta(s)\big)\big]$
  - like in Double Q-learning (Lecture 5)
- CDQ: two critics against overestimation
  - minimum, *unlike* in Double Q-learning!
- TD3: regularization with clipped noise
  $\boldsymbol{\epsilon} \sim \text{clip}\big(\mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2), -\boldsymbol{c}, \boldsymbol{c}\big) \in \mathbb{R}^{|\mathcal{A}|}$



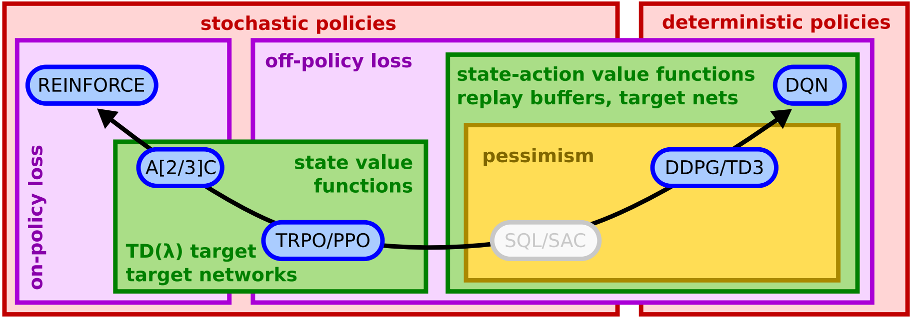
$$\mathcal{L}_i^Q[\phi_i] \quad = \quad \mathbb{E}_\mu\Bigg[\sum_{t=0}^{n-1}\Big(r_t + \gamma \min_{j\in\{1,2\}} Q_{\phi'_j}\big(s_{t+1}, \boldsymbol{\pi}_{\theta'}(s_{t+1}) + \boldsymbol{\epsilon}\big) - Q_{\phi_i}(s_t, a_t)\Big)^2\Bigg]$$



images and results from Fujimoto et al. (TD3, 2018)

DDPG/TD3 in comparison

- Only for deterministic policies and continuous actions
  - allows sample efficient off-policy sampling from replay buffer
  - state-action Q-value functions generalize worse
  - TD3 stabilizes Q-value with pessimism



TD3 is one of the highest performing algorithms for continuous actions

7.1 Summary

- DPG learns deterministic policies for continuous actions
- Requires Q-value of current policy function as critic
- Implementation with two optimizers
- TD3 is a modern version based on pessimism and action noise

Learning Objectives

LO7.1: Derive and implement the DPG losses
LO7.2: Explain the difference between DDPG and TD3

7.2

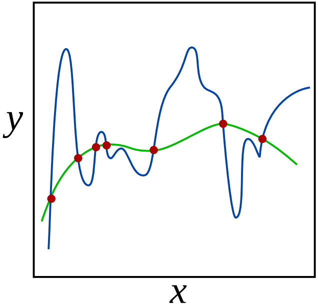
Off-Policy Actor-Critic

Robust decision making

TUDelft

7.2 Regularization

- TD3 used noisy actions as regularization of the policy
- Regularization smoothes functions by lowering ℓ, e.g. $\|\boldsymbol{a}\|$
  - or averages over ensemble like Dropout
  - data augmentation sometimes called regularization

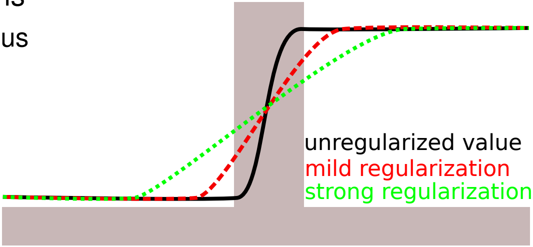


For regularization and generalization see Lecture 4

image source: www.wikipedia.org

7.2 Regularization

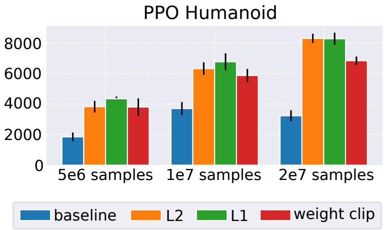
- TD3 used noisy actions as regularization of the policy
- Regularization smoothes functions by lowering ℓ, e.g. $\|\boldsymbol{a}\|$
  - or averages over ensemble like Dropout
  - data augmentation sometimes called regularization
- No effect for value functions
  - smooth values dangerous
  - e.g. near walls



effect of regularization still open question, see Liu et al. (2021, image source) for details on presented example

7.2 Regularization

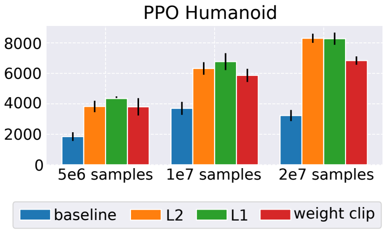
- TD3 used noisy actions as regularization of the policy
- Regularization smoothes functions by lowering ℓ, e.g. $\|\boldsymbol{a}\|$
  - or averages over ensemble like Dropout
  - data augmentation sometimes called regularization
- No effect for value functions
  - smooth values dangerous
  - e.g. near walls
- Measurable effect for policies
  - why does this help?



effect of regularization still open question, see Liu et al. (2021, image source) for details on presented example

Regularization

- TD3 used noisy actions as regularization of the policy
- Regularization smoothes functions by lowering ℓ, e.g. $\|\boldsymbol{a}\|$
  - or averages over ensemble like Dropout
  - data augmentation sometimes called regularization
- No effect for value functions
  - smooth values dangerous
  - e.g. near walls
- Measurable effect for policies
  - why does this help?
  - regularized = smooth = stochastic
  - why are stochastic policies good?



effect of regularization still open question, see Liu et al. (2021, image source) for details on presented example

7.2 Robust reinforcement learning

- Approximation introduces local errors in the policy
  - online sampling corrects mistakes slowly
  - long-term consequences of small errors varies
  - robust decisions should have robust consequences
  - can improve learning even in deterministic MDP

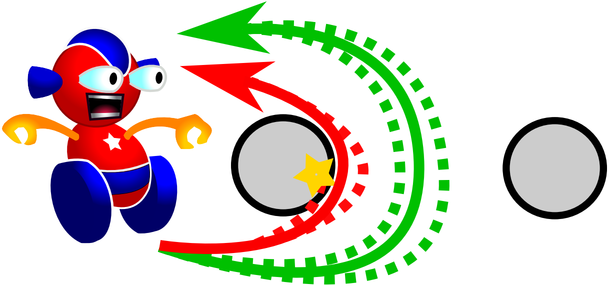


image source freesvg.org

7.2 Robust reinforcement learning

- Approximation introduces local errors in the policy
  - online sampling corrects mistakes slowly
  - long-term consequences of small errors varies
  - robust decisions should have robust consequences
  - can improve learning even in deterministic MDP
- Robust RL disrupts MDP with term $\delta \in \Delta$
  - by changing transitions or rewards, e.g. $P(s'|s,a,\delta) := P(s'+\delta|s,a)$
  - withstanding disruption δ is rewarded, e.g. $\bar{r}(\delta) := \alpha\|\delta\|$

$$V^*_{\mathsf{R}}(s) \quad := \quad \max_{a \in \mathcal{A}} \min_{\delta \in \Delta} \Big(r(s,a,\delta) + \bar{r}(\delta) + \gamma \int P(s'|s,a,\delta)\, V^*_{\mathsf{R}}(s')\, ds' \Big)$$

basic concept by Morimoto and Doya (2001), see `slides` by Marek Petrik and the recent review by Moos et al. (2022) for details

7.2 Robust reinforcement learning

- Approximation introduces local errors in the policy
  - online sampling corrects mistakes slowly
  - long-term consequences of small errors varies
  - robust decisions should have robust consequences
  - can improve learning even in deterministic MDP
- Robust RL disrupts MDP with term $\delta \in \Delta$
  - by changing transitions or rewards, e.g. $P(s'|s,a,\delta) := P(s'+\delta|s,a)$
  - withstanding disruption δ is rewarded, e.g. $\bar{r}(\delta) := \alpha\|\delta\|$

$$V^*_{\mathrm{R}}(s) \quad := \quad \max_{a\in\mathcal{A}} \min_{\delta\in\Delta} \Big(r(s,a,\delta) + \bar{r}(\delta) + \gamma \int P(s'|s,a,\delta)\, V^*_{\mathrm{R}}(s')\, ds' \Big)$$

- Requires controllable amounts of noise/disruptions δ
  - can we motivate $\bar{r}(\delta)$ without such control?
  - e.g. how much noise should be added to actions in TD3?

basic concept by Morimoto and Doya (2001), see `slides` by Marek Petrik and the recent review by Moos et al. (2022) for details

7.2 Planning as inference I

- Probability of trajectory $\tau_n = [s_0, a_0, \ldots, s_n]$ under prior policy μ

$$p^\mu(\tau_n) \quad = \quad \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t, a_t)$$

(PAI, Levine, 2018)

7.2 Planning as inference I

- Probability of trajectory $\tau_n = [s_0, a_0, \dots, s_n]$ under prior policy μ
  - introducing "optimality variables" $O_t \in \{0, 1\}$

$$p^\mu(\tau_n) = \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t, a_t)$$

$$p^\mu(\tau_n, \{O_t\}_{t=0}^{n-1}) = \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, p(O_t|s_t, a_t)\, P(s_{t+1}|s_t, a_t)$$

(PAI, Levine, 2018)

7.2 Planning as inference I

- Probability of trajectory $\tau_n = [s_0, a_0, \ldots, s_n]$ under prior policy μ
  - introducing "optimality variables" $O_t \in \{0, 1\}$
  - probability that all O_t are "optimal"

$$
\begin{aligned}
p^\mu(\tau_n) &= \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t, a_t) \\
p^\mu(\tau_n, \{O_t\}_{t=0}^{n-1}) &= \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, p(O_t|s_t, a_t)\, P(s_{t+1}|s_t, a_t) \\
p^\mu(\tau_n|\{O_t{=}1\}_{t=0}^{n-1}) &\propto \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, p(O_t{=}1|s_t, a_t)\, P(s_{t+1}|s_t, a_t)
\end{aligned}
$$

(PAI, Levine, 2018)

7.2 Planning as inference I

- Probability of trajectory $\tau_n = [s_0, a_0, \ldots, s_n]$ under prior policy μ
  - introducing "optimality variables" $O_t \in \{0, 1\}$
  - probability that all O_t are "optimal"
  - define optimality $p(O_t{=}1|s_t, a_t) \propto \exp\big(\frac{1}{\alpha} r(s_t, a_t)\big)$

$$
\begin{aligned}
p^\mu(\tau_n) &= \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t, a_t) \\
p^\mu(\tau_n, \{O_t\}_{t=0}^{n-1}) &= \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, p(O_t|s_t, a_t)\, P(s_{t+1}|s_t, a_t) \\
p^\mu(\tau_n|\{O_t{=}1\}_{t=0}^{n-1}) &\propto \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, p(O_t{=}1|s_t, a_t)\, P(s_{t+1}|s_t, a_t) \\
p^\mu(\tau_n|\{O_t{=}1\}_{t=0}^{n-1}) &\propto \rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t) \exp\big(\tfrac{1}{\alpha} r(s_t, a_t)\big)\, P(s_{t+1}|s_t, a_t)
\end{aligned}
$$

(PAI, Levine, 2018)

7.2 Planning as inference II

- $p^{\mu}(\tau_n | \{O_t = 1\}_{t=0}^{n-1})$ is the distribution of "optimal" trajectories!
  - Minimize KL-divergence between $p^{\pi_\theta}(\tau_n)$ and $p^{\mu}(\tau_n | \{O_t = 1\}_{t=0}^{n-1})$

$$\mathcal{L}_\pi^{\text{PAI}} \quad := \quad D_{\text{KL}}\left[p^{\pi_\theta}(\tau_n) \,\middle\|\, p^{\mu}(\tau_n | \{O_t = 1\}_{t=0}^{n-1})\right]$$

(PAI, Levine, 2018)

7.2 Planning as inference II

- $p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$ is the distribution of "optimal" trajectories!
  - Minimize KL-divergence between $p^{\pi_\theta}(\tau_n)$ and $p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$

$$
\begin{aligned}
\mathcal{L}_\pi^{\text{PAI}} \quad &:= \quad D_{\text{KL}}\Big[p^{\pi_\theta}(\tau_n) \,\Big\|\, p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})\Big] \\
&\propto \quad \int p^{\pi_\theta}(\tau_n) \ln \frac{\rho(s_0)\prod_{t=0}^{n-1} \pi_\theta(a_t|s_t)\, P(s_{t+1}|s_t,a_t)}{\rho(s_0)\prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t,a_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\, d\tau_n
\end{aligned}
$$

(PAI, Levine, 2018)

7.2 Planning as inference II

- $p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$ is the distribution of "optimal" trajectories!
  - Minimize KL-divergence between $p^{\pi_\theta}(\tau_n)$ and $p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$

$$
\begin{aligned}
\mathcal{L}_\pi^{\text{PAI}} &:= D_{\text{KL}}\Big[p^{\pi_\theta}(\tau_n) \,\Big\|\, p^\mu(\tau_n|\{O_t = 1\}_{t=0}^{n-1})\Big] \\
&\propto \int p^{\pi_\theta}(\tau_n) \ln \frac{\rho(s_0)\prod_{t=0}^{n-1} \pi_\theta(a_t|s_t)\, P(s_{t+1}|s_t,a_t)}{\rho(s_0)\prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t,a_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\, d\tau_n \\
&= \int p^{\pi_\theta}(\tau_n) \ln\Big(\prod_{t=0}^{n-1} \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\Big)\, d\tau_n
\end{aligned}
$$

(PAI, Levine, 2018)

7.2 Planning as inference II

- $p^{\mu}(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$ is the distribution of "optimal" trajectories!
  - Minimize KL-divergence between $p^{\pi_\theta}(\tau_n)$ and $p^{\mu}(\tau_n|\{O_t = 1\}_{t=0}^{n-1})$

$$
\begin{aligned}
\mathcal{L}_\pi^{\text{PAI}} &:= D_{\text{KL}}\Big[p^{\pi_\theta}(\tau_n) \,\Big\|\, p^{\mu}(\tau_n|\{O_t = 1\}_{t=0}^{n-1})\Big] \\
&\propto \int p^{\pi_\theta}(\tau_n) \ln \frac{\rho(s_0) \prod_{t=0}^{n-1} \pi_\theta(a_t|s_t)\, P(s_{t+1}|s_t,a_t)}{\rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t,a_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\, d\tau_n \\
&= \int p^{\pi_\theta}(\tau_n) \ln\Big(\prod_{t=0}^{n-1} \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\Big)\, d\tau_n \\
&= -\int p^{\pi_\theta}(\tau_n) \sum_{t=0}^{n-1} \Big(\frac{1}{\alpha} r(s_t,a_t) - \ln \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}\Big) d\tau_n
\end{aligned}
$$

(PAI, Levine, 2018)

Planning as inference II

- $p^\mu(\tau_n | \{O_t = 1\}_{t=0}^{n-1})$ is the distribution of "optimal" trajectories!
  - Minimize KL-divergence between $p^{\pi_\theta}(\tau_n)$ and $p^\mu(\tau_n | \{O_t = 1\}_{t=0}^{n-1})$
  - using the uniform policy as prior μ yields entropy $\mathcal{H}$

$$
\begin{aligned}
\mathcal{L}_\pi^{\text{PAI}} &:= D_{\text{KL}}\Big[p^{\pi_\theta}(\tau_n) \,\Big\|\, p^\mu(\tau_n | \{O_t = 1\}_{t=0}^{n-1})\Big] \\
&\propto \int p^{\pi_\theta}(\tau_n) \ln \frac{\rho(s_0) \prod_{t=0}^{n-1} \pi_\theta(a_t|s_t)\, P(s_{t+1}|s_t,a_t)}{\rho(s_0) \prod_{t=0}^{n-1} \mu(a_t|s_t)\, P(s_{t+1}|s_t,a_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))}\, d\tau_n \\
&= \int p^{\pi_\theta}(\tau_n) \ln \Big(\prod_{t=0}^{n-1} \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)\, \exp(\frac{1}{\alpha} r(s_t,a_t))} \Big)\, d\tau_n \\
&= -\int p^{\pi_\theta}(\tau_n) \sum_{t=0}^{n-1} \Big(\frac{1}{\alpha} r(s_t, a_t) - \ln \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)} \Big)\, d\tau_n \\
&\equiv -\mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{n-1} \Big(r(s_t, a_t) + \alpha \mathcal{H}\big[\pi_\theta(\cdot|s_t)\big] \Big) \Big]
\end{aligned}
$$

- Stochastic policy gradient with additional entropy reward!

maximum entropy RL (PAI, Levine, 2018) is a lower bound to robust RL (as proven in Eysenbach and Levine, 2021)

Stochastic vs. determinisitic policy gradients

- Stochastic policy $\pi_\theta(a|s)$ from family $\hat{\pi}(a|\boldsymbol{f}_\theta(s))$
  - $\boldsymbol{f}_\theta(s) \in \mathbb{R}^m$ denotes the sufficient statistics at $s \in \mathcal{S}$
  - e.g. $\hat{\pi}(a|\boldsymbol{f}_\theta(s)) = \mathcal{N}(a \,|\, \mu_\theta(s), \sigma^2_\theta(s))$ with $\boldsymbol{f}_\theta(s) = [\mu_\theta(s), \sigma^2_\theta(s)]$

$$\nabla_\theta \mathcal{L}_\pi[\theta] \quad \approx \quad -\sum_{t=0}^{n-1} \gamma^t \iint \xi_t^\mu(s) \, \nabla_\theta \underbrace{\pi_\theta(a|s)}_{\hat{\pi}(a|\boldsymbol{f}_\theta(s))} \, Q^\pi(s,a) \, da \, ds$$

here we use $Q^\pi(s,a) = \mathbb{E}_\pi[R_0|^{s_0=s}_{a_0=a}]$, and the approximation drops $\frac{\xi_t^\pi(s_t)}{\xi^\mu(s_t)}$, see Sutton et al. (2016) for ways to estimate it

Stochastic vs. determinisitic policy gradients

- Stochastic policy $\pi_\theta(a|s)$ from family $\hat{\pi}(a|\boldsymbol{f}_\theta(s))$
  - $\boldsymbol{f}_\theta(s) \in \mathbb{R}^m$ denotes the sufficient statistics at $s \in \mathcal{S}$
  - e.g. $\hat{\pi}(a|\boldsymbol{f}_\theta(s)) = \mathcal{N}(a \,|\, \mu_\theta(s), \sigma^2_\theta(s))$ with $\boldsymbol{f}_\theta(s) = [\mu_\theta(s), \sigma^2_\theta(s)]$

$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\pi[\theta] \quad &\approx \quad -\sum_{t=0}^{n-1} \gamma^t \iint \xi_t^\mu(s)\, \nabla_\theta \underbrace{\pi_\theta(a|s)}_{\hat{\pi}(a|\boldsymbol{f}_\theta(s))} \, Q^\pi(s,a)\, da\, ds \\
&= \quad -\sum_{t=0}^{n-1} \gamma^t \int \xi_t^\mu(s)\, \nabla_\theta \underbrace{\boldsymbol{f}_\theta(s)}_{\boldsymbol{\pi}_\theta(s)} \, \nabla_{\boldsymbol{f}} \underbrace{\int \hat{\pi}(a|\boldsymbol{f})\, Q^\pi(s,a)\, da}_{\hat{Q}^{\boldsymbol{\pi}}(s,\boldsymbol{f})} \Big|_{\boldsymbol{f}=\boldsymbol{f}_\theta(s)} ds
\end{aligned}
$$

Ciosek and Whiteson (2018) developed this view, called expected policy gradients (EPG)

Stochastic vs. determinisitic policy gradients

- Stochastic policy $\pi_\theta(a|s)$ from family $\hat{\pi}(a|\boldsymbol{f}_\theta(s))$
  - $\boldsymbol{f}_\theta(s) \in \mathbb{R}^m$ denotes the sufficient statistics at $s \in \mathcal{S}$
  - e.g. $\hat{\pi}(a|\boldsymbol{f}_\theta(s)) = \mathcal{N}(a \,|\, \mu_\theta(s), \sigma^2_\theta(s))$ with $\boldsymbol{f}_\theta(s) = [\mu_\theta(s), \sigma^2_\theta(s)]$

$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\pi[\theta] \quad &\approx \quad -\sum_{t=0}^{n-1} \gamma^t \iint \xi^\mu_t(s)\, \nabla_\theta \underbrace{\pi_\theta(a|s)}_{\hat{\pi}(a|\boldsymbol{f}_\theta(s))}\, Q^\pi(s,a)\, da\, ds \\
&= \quad -\sum_{t=0}^{n-1} \gamma^t \int \xi^\mu_t(s)\, \nabla_\theta \underbrace{\boldsymbol{f}_\theta(s)}_{\boldsymbol{\pi}_\theta(s)}\, \nabla_{\boldsymbol{f}} \underbrace{\int \hat{\pi}(a|\boldsymbol{f})\, Q^\pi(s,a)\, da}_{\hat{Q}^{\boldsymbol{\pi}}(s,\boldsymbol{f})} \Big|_{\boldsymbol{f}=\boldsymbol{f}_\theta(s)} ds \\
&= \quad -\mathbb{E}_\mu\Big[\sum_{t=0}^{n-1} \gamma^t \nabla_\theta \hat{Q}^{\boldsymbol{\pi}}\big(s_t, \boldsymbol{\pi}_\theta(s_t)\big)\Big]
\end{aligned}
$$

- SPG for family of distributions $\equiv$ DPG on sufficient statistics!

Ciosek and Whiteson (2018) developed this view, called expected policy gradients (EPG)

7.2 The reparametrization trick

- Problem: estimate $\nabla_\theta \hat{Q}^{\boldsymbol{\pi}}\big(s, \boldsymbol{f}_\theta(s)\big) = \int \nabla_\theta \hat{\pi}\big(\boldsymbol{a} | \boldsymbol{f}_\theta(s)\big)\, Q^\pi(s, \boldsymbol{a})\, d\boldsymbol{a}$
  - hard to solve in general

this trick will be used in SQL on slide 19 and SAC on slide 20

The reparametrization trick

- Problem: estimate $\nabla_\theta \hat{Q}^{\boldsymbol{\pi}}\big(s, \boldsymbol{f}_\theta(s)\big) = \int \nabla_\theta \hat{\pi}\big(\boldsymbol{a} | \boldsymbol{f}_\theta(s)\big)\, Q^\pi(s, \boldsymbol{a})\, d\boldsymbol{a}$
  - hard to solve in general
- Bijective mappings can model arbitrary distributions
  - $\boldsymbol{a} := \boldsymbol{f}_\theta(s, \boldsymbol{\epsilon}) \sim \pi_\theta(\cdot|s)\,, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I}),$ for bijective $\boldsymbol{f}_\theta$

$$\begin{aligned}
\nabla_\theta \hat{Q}^{\boldsymbol{\pi}}\big(s, \boldsymbol{f}_\theta(s)\big) &= \nabla_\theta \mathbb{E}\Big[Q^\pi(s, \boldsymbol{a}) \,\Big|\, \boldsymbol{a} \sim \pi_\theta(\cdot|s)\Big] \\
&= \mathbb{E}\Big[\nabla_\theta Q^\pi\big(s, \boldsymbol{f}_\theta(s, \boldsymbol{\epsilon})\big) \,\Big|\, \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})\Big] \\
&\approx \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} Q^\pi\big(s, \boldsymbol{f}_\theta(s, \boldsymbol{\epsilon}_i)\big)\,, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})
\end{aligned}$$

- One forward pass over batch with i.i.d. $\boldsymbol{\epsilon}_i$

this trick will be used in SQL on slide 19 and SAC on slide 20

7.2 Summary

- Robust RL chooses *disruption* that minimizes return
- Related planning as inference (PAI) is RL with entropy reward

$\Rightarrow$ Stochastic policies allow robust control!

- SPG $\equiv$ DPG on sufficient statistics
- Reparametrization trick to compute SPG with DPG

Learning Objectives

LO7.3: Explain robust RL and planning as inference
LO7.4: Explain the connection between SPG and DPG
LO7.5: Derive the reparametrization trick

7.3

Off-Policy Actor-Critic

Maximum entropy RL

Maximum entropy RL

- We can use additional entropy reward for policy gradient
- Or we can try to maximize the *soft* Q*-values* directly:

$$Q^\pi_{\text{soft}}(s,a) \;:=\; r(s,a) + \mathbb{E}_\pi\Big[\sum_{t=1}^{\infty}\gamma^t\Big(r_t + \alpha\mathcal{H}\big[\pi(\cdot|s_t)\big]\Big)\Big|\, {}^{s_0=s}_{a_0=a}\Big]$$

$$Q^*_{\text{soft}}(s,a) \;:=\; \max_\pi Q^\pi_{\text{soft}}(s,a) \;=\; r(s,a) + \gamma\mathbb{E}\big[V^*_{\text{soft}}(s')\,|\, s'\sim P(\cdot|s,a)\big]$$

assignment sheet 3

7.3 Maximum entropy RL

- We can use additional entropy reward for policy gradient
- Or we can try to maximize the *soft Q-values* directly:

$$Q^\pi_{\text{soft}}(s,a) \;:=\; r(s,a) + \mathbb{E}_\pi\Big[\sum_{t=1}^\infty \gamma^t \Big(r_t + \alpha \mathcal{H}\big[\pi(\cdot|s_t)\big]\Big) \Big| \begin{smallmatrix} s_0=s \\ a_0=a \end{smallmatrix}\Big]$$

$$Q^*_{\text{soft}}(s,a) \;:=\; \max_\pi Q^\pi_{\text{soft}}(s,a) \;=\; r(s,a) + \gamma \mathbb{E}\big[V^*_{\text{soft}}(s') \,\big|\, s' \sim P(\cdot|s,a)\big]$$

- There exist an analytical solution for V^*_{soft} and π^*_{soft}:

$$V^*_{\text{soft}}(s) \;:=\; \alpha \ln\Big(\int \exp\big(\tfrac{1}{\alpha} Q^*_{\text{soft}}(s,a)\big)\, da\Big) \;\equiv\; \alpha \operatorname*{softmax}_{a \in \mathcal{A}} \tfrac{1}{\alpha} Q^*_{\text{soft}}(s,a)$$

$$\pi^*_{\text{soft}}(a|s) \;:=\; \arg\max_\pi Q^\pi_{\text{soft}}(s,a) \;=\; \exp\Big(\tfrac{1}{\alpha} Q^*_{\text{soft}}(s,a) - \tfrac{1}{\alpha} V^*_{\text{soft}}(s)\Big)$$

$\max_a f(a) = \lim_{\alpha \to 0} \alpha \operatorname*{softmax}_a \frac{1}{\alpha} f(a)$, proof for V^*_{soft} and π^*_{soft} in Ziebart (2010) and/or Haarnoja et al. (2017, 2018a)

7.3 Soft Q-learning (SQL)

- One of the first attempts at maximum entropy RL
- SQL optimizes Q_ϕ and a corresponding policy π_θ
  - minimize KL-divergence between π_θ and π^*_{soft}
  - $V_\phi(s) := \alpha \ln \mathbb{E}\big[\frac{1}{\mu(a)} \exp\big(\frac{1}{\alpha} Q_\phi(s,a)\big) \big| a \sim \mu\big]$

$$
\begin{aligned}
\mathcal{L}_Q^{\text{soft}}[\phi] \quad &:= \quad \mathbb{E}\Big[\big(r + \gamma V_{\phi'}(s') - Q_\phi(s,a)\big)^2 \Big| \langle s,a,r,s'\rangle \sim \mathcal{D}\Big] \\
\mathcal{L}_\pi^{\text{soft}}[\theta] \quad &:= \quad \mathbb{E}\Big[D_{\text{KL}}\big[\pi_\theta(\cdot|s) \big\| \underbrace{\exp\big(\tfrac{1}{\alpha} Q_\phi(s,\cdot) - \tfrac{1}{\alpha} V_\phi(s)\big)}_{\pi^*_{\text{soft}}(\cdot|s)}\big] \Big| s \sim \mathcal{D}\Big]
\end{aligned}
$$

- SQL minimizes $\mathcal{L}_\pi^{\text{soft}}$ with *Stein variational gradient descent*
  - complicated and soon supplanted by SAC

(SQL, Haarnoja et al., 2017)

7.3 Soft actor-critic (SAC)

- Same basic losses as SQL, but with a ton of tricks
  - extra state-value function $V_\psi(s)$
  - TD3 style twin Q-values $\bar{Q}(s,a) := \min\limits_{i\in\{1,2\}} Q_{\phi_i}(s,a)$

$$
\begin{aligned}
\mathcal{L}_V^{\text{soft}}[\psi] &:= \mathbb{E}\Big[\Big(\mathbb{E}[\bar{Q}(s,a) - \alpha \ln \pi_\theta(a|s) \,\big|\, {\scriptstyle a\sim\pi_\theta(\cdot|s)}] - V_\psi(s)\Big)^2 \,\Big|\, {\scriptstyle s\sim\mathcal{D}}\Big] \\
\mathcal{L}_Q^{\text{soft}}[\phi] &:= \mathbb{E}\Big[\big(r + \gamma V_{\psi'}(s') - Q_\phi(s,a)\big)^2 \Big| \, {\scriptstyle \langle s,a,r,s'\rangle\sim\mathcal{D}}\Big] \\
\mathcal{L}_\pi^{\text{soft}}[\theta] &:= \mathbb{E}\Big[D_{\text{KL}}\big[\pi_\theta(\cdot|s) \big\| \exp\big(\tfrac{1}{\alpha}\bar{Q}(s,\cdot) - \tfrac{1}{\alpha}V_\psi(s)\big)\big] \Big| \, {\scriptstyle s\sim\mathcal{D}}\Big]
\end{aligned}
$$

- SAC implementations branch off TD3!
  - uses replay buffer, pessimism, but not clipped noise
  - stochastic policy via reparametrization trick (slide 16)
  - action squashing for bounded action spaces
  - automatic entropy adjustment

spinningup.openai.com/en/latest/algorithms/sac.html

(SAC, Haarnoja et al., 2018a,b)

7.3 Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0},\mathbf{I})$
  - "squash" action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon})\big)$

- How does this change $\ln \pi(\boldsymbol{a}|s)$ for the entropy?

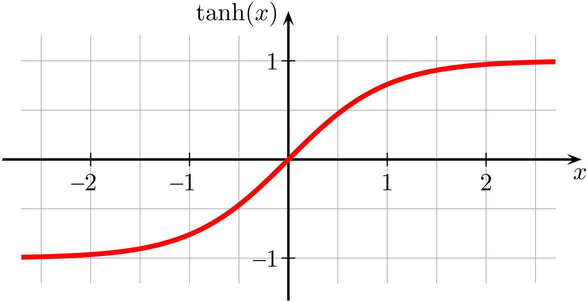


image source wikipedia.org

7.3 Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})$
  - "squash" action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon})\big)$
- How does this change $\ln \pi(\boldsymbol{a}|s)$ for the entropy?
  - $\pi(\boldsymbol{a}|s) = \mathcal{N}\Big(\tanh^{-1}(\boldsymbol{a}) \,\big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\Big) \left|\det\left[\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}}\right]\right|$

https://en.wikipedia.org/wiki/Probability_density_function#Vector_to_vector

7.3 Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})$
  - "squash" action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon})\big)$

- How does this change $\ln \pi(\boldsymbol{a}|s)$ for the entropy?
  - $\pi(\boldsymbol{a}|s) = \mathcal{N}\Big(\tanh^{-1}(\boldsymbol{a}) \,\Big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\Big) \left|\det\left[\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}}\right]\right|$
  - inverse function th.: $\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}} = \left(\frac{\partial \tanh(\boldsymbol{u})}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{u}=\tanh^{-1}(\boldsymbol{a})}\right)^{-1}$

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

7.3 Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})$
  - "squash" action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s, \boldsymbol{\epsilon})\big)$
- How does this change $\ln \pi(\boldsymbol{a}|s)$ for the entropy?
  - $\pi(\boldsymbol{a}|s) = \mathcal{N}\Big(\tanh^{-1}(\boldsymbol{a}) \,\big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\Big) \left|\det\left[\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}}\right]\right|$
  - inverse function th.: $\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}} = \left(\frac{\partial \tanh(\boldsymbol{u})}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{u}=\tanh^{-1}(\boldsymbol{a})}\right)^{-1}$
  - determinant of inverse matrix: $\det \mathbf{J}^{-1} = (\det \mathbf{J})^{-1}$

https://en.wikipedia.org/wiki/Determinant#Multiplicativity_and_matrix_groups

7.3 Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0},\mathbf{I})$
  - “squash” action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon})\big)$
- How does this change $\ln \pi(\boldsymbol{a}|s)$ for the entropy?
  - $\pi(\boldsymbol{a}|s) = \mathcal{N}\Big(\tanh^{-1}(\boldsymbol{a}) \,\big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\Big) \left|\det\left[\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}}\right]\right|$
  - inverse function th.: $\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}} = \left(\frac{\partial \tanh(\boldsymbol{u})}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{u}=\tanh^{-1}(\boldsymbol{a})}\right)^{-1}$
  - determinant of inverse matrix: $\det \mathbf{J}^{-1} = (\det \mathbf{J})^{-1}$
  - diagonal Jacobian matrix $J_{kk} := \frac{\partial \tanh(u_k)}{\partial u_k} = \big(1 - \tanh^2(u_k)\big)$

    $\Rightarrow \quad \det \mathbf{J} = \prod_k J_{kk}$

https://en.wikipedia.org/wiki/Determinant#Immediate_consequences

Action squashing

- Gaussian distribution not suitable for bounded action spaces
  - use Gaussian $\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon}) := \boldsymbol{\mu}_\theta(s) + \boldsymbol{\sigma}^2_\theta(s) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\cdot|\mathbf{0},\mathbf{I})$
  - “squash” action into (-1,1): $\boldsymbol{a} := \tanh\big(\boldsymbol{f}_\theta(s,\boldsymbol{\epsilon})\big)$
- How does this change $\ln\pi(\boldsymbol{a}|s)$ for the entropy?
  - $\pi(\boldsymbol{a}|s) = \mathcal{N}\Big(\tanh^{-1}(\boldsymbol{a}) \,\Big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\Big) \left|\det\left[\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}}\right]\right|$
  - inverse function th.: $\frac{\partial \tanh^{-1}(\boldsymbol{a}')}{\partial \boldsymbol{a}'}\Big|_{\boldsymbol{a}'=\boldsymbol{a}} = \left(\frac{\partial \tanh(\boldsymbol{u})}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{u}=\tanh^{-1}(\boldsymbol{a})}\right)^{-1}$
  - determinant of inverse matrix: $\det \mathbf{J}^{-1} = (\det \mathbf{J})^{-1}$
  - diagonal Jacobian matrix $J_{kk} := \frac{\partial \tanh(u_k)}{\partial u_k} = \big(1 - \tanh^2(u_k)\big)$
    $\Rightarrow \quad \det \mathbf{J} = \prod_k J_{kk}$
- $$\ln\pi(\boldsymbol{a}|s) = \underbrace{\ln\mathcal{N}\big(\tanh^{-1}(\boldsymbol{a}) \,\big|\, \boldsymbol{\mu}_\theta(s), \boldsymbol{\sigma}^2_\theta(s)\big)}_{\text{Gaussian entropy}} - \underbrace{\textstyle\sum_k \ln\big(1 - \tanh(a_k)\big)}_{\text{correction term for squashing}}$$

action squashing can be numerically unstable

see also Haarnoja et al. (2018b, Appendix C)

7.3 Automatic entropy adjustment

- Entropy hyper-parameter α is hard to choose
  - too low and the entropy term is insignificant
  - too high and the reward is insignificant

- Automatic adjustment by defining target entropy H

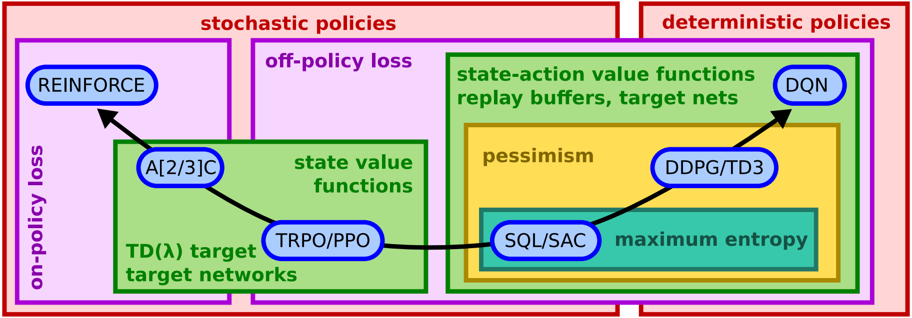
$$\min_{\theta} \mathcal{L}_\pi[\theta] \qquad \text{s.t.} \qquad -\mathbb{E}\Big[\ln \pi_\theta(a|s) \,\Big|\, {}_{a\sim\pi_\theta(\cdot|s)}^{s\sim\mathcal{D}}\Big] \geq H$$

- Gradient descend w.r.t. dual variable α of the Lagrangian
  - averaged over minibatch $\{s_t\}_{i=1}^n$ and $a_t^i \sim \pi_\theta(\cdot|s_t), \forall i, t$
  - $\alpha \leftarrow \alpha + \eta\, \mathbb{E}\Big[\ln \pi_\theta(a|s) + H \,\Big|\, {}_{a\sim\pi_\theta(\cdot|s)}^{s\sim\mathcal{D}}\Big] \approx \alpha + \frac{\eta}{nm} \sum_{t=1}^{n}\sum_{i=1}^{m} \ln \pi_\theta(a_t^i|s_t) + \eta H$

based on derivation in Haarnoja et al. (2018b)

SQL/SAC in comparison

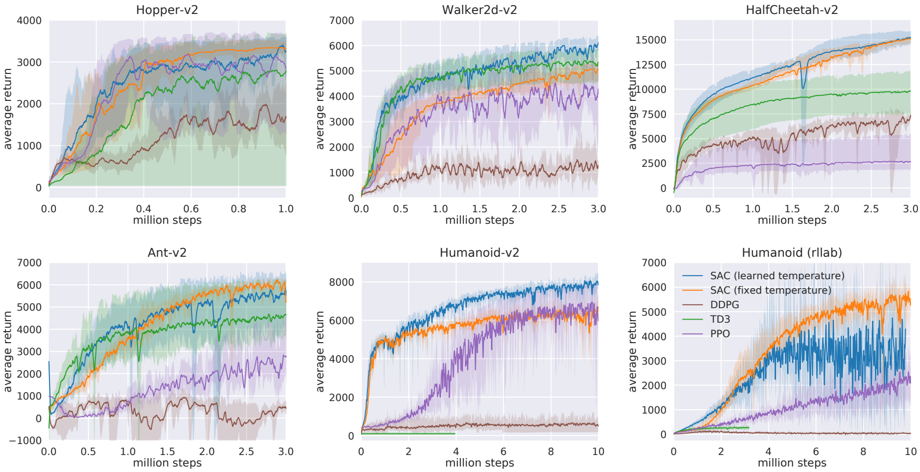
- Entropy maximization for robustness and exploration
  - reparameterization trick to extend TD3 to stochastic policies
  - state-action Q-value functions generalize worse (like TD3)
  - SAC and TD3 stabilize Q-value with pessimism



SAC is also one of the best RL algorithms, explores slightly better than TD3 and is very common in *simulated* robotics

7.3 Performance comparison

- AC2/3 $<$ DDPG $<$ PPO $\overset{?}{\leq}$ (TD3 $\overset{?}{\approx}$ SAC) (no consensus)

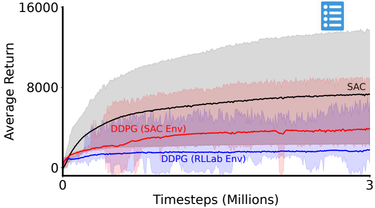


results from SAC paper (Haarnoja et al., 2018b), for results that favor TD3, see slide 8

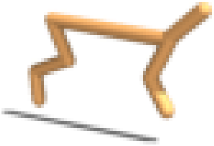
7.3 Policy improvement in DRL algorithms

- Pure on-policy algorithms (slow, stable, require state-value v_ϕ)
  - Reinforce $\mathcal{L}^{\text{RE}}_\pi[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t R_t \ln \pi_\theta(a_t|s_t)$ (high variance)
  - A(|2|3)C $\mathcal{L}^{\text{AC}}_\pi[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t \underbrace{\Big(y_t^{n/\lambda}(\tau_n) - V_\phi(s_t)\Big)}_{A_t} \ln \pi_\theta(a_t|s_t)$
- On-off-policy algorithms (fast, need trust-region for stability)
  - Off-PAC $\mathcal{L}^{\text{OPAC}}_\mu[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t A_t \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}$ (extremely unstable)
  - TRPO $\mathcal{L}^{\text{TRPO}}_\mu[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t A_t \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}$ s.t. $\frac{1}{n}\sum_{t=1}^{n} D_{\text{KL}}\big(\mu(\cdot|s_t)\|\pi_\theta(\cdot|s_t)\big)$
  - PPO $\mathcal{L}^{\text{PPO}}_\mu[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t \min\Big(A_t \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}, A_t \,\text{clip}\big(\frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}, 1-\epsilon, 1+\epsilon\big)\Big)$
- Off-policy algorithms (fast, require Q-value, easily overestimated)
  - DDPG/TD3 $\mathcal{L}^{\text{DDPG}}_\mu[\theta] = -\frac{1}{n}\sum_{t=1}^{n}\gamma^t \hat{Q}^{\boldsymbol{\pi}}_\phi\big(s_t, \boldsymbol{\pi}(s_t)\big)$ (Q-value of suff. statistics)
  - SAC $\mathcal{L}^{\text{soft}}_\pi[\theta] = \frac{1}{n}\sum_{t=1}^{n} D_{\text{KL}}\Big[\pi_\theta(\cdot|s)\Big\|\exp\big(\frac{1}{\alpha}\min_i Q_{\phi_i}(s_t,\cdot) - \frac{1}{\alpha}V_\psi(s_t)\big)\Big]$
  - DQN $\pi(a|s_t) = \arg\max_a Q^*_\phi(s_t, a)$ (greedy Q-values Q^*)

7.3 A word of caution



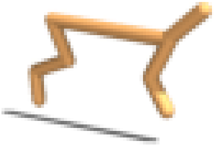
- Replicating SAC results (Haarnoja et al., 2018a)
  - Mujoco's Half-Cheetha env
  - very unreliable training [30 seeds]

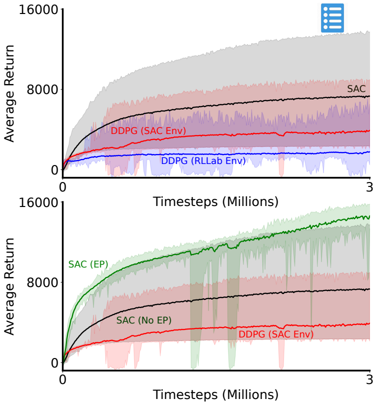


figures and story are from Patterson et al. (2023), which is a good take on best practices in DRL

7.3 A word of caution

- Replicating SAC results (Haarnoja et al., 2018a)
- Found undocumented feature
  - random initial “burn-in” phase (EP)
  - performance now reliable

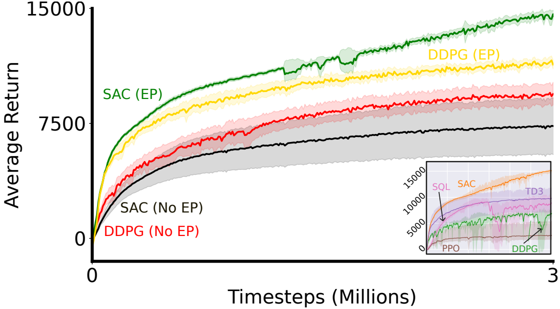


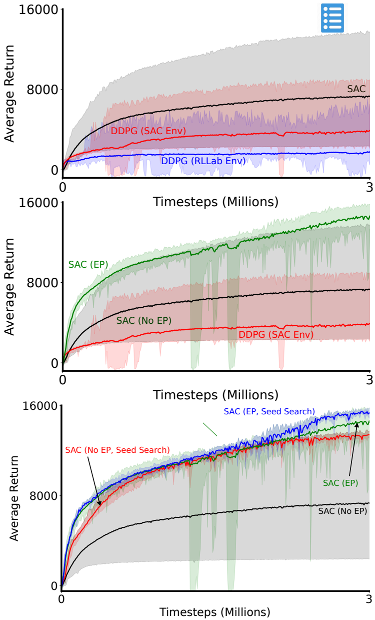


figures and story are from Patterson et al. (2023), which is a good take on best practices in DRL

7.3 A word of caution

- Replicating SAC results (Haarnoja et al., 2018a)
- Found undocumented feature
- Cherry-picking seeds (**don't do this!**)
  - same performance as (Haarnoja et al., 2018a)

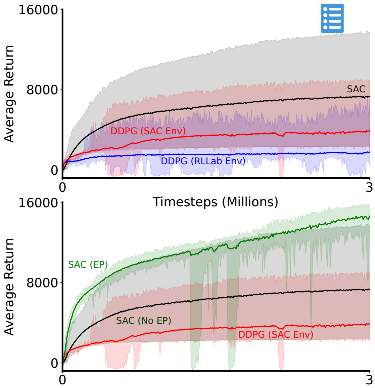


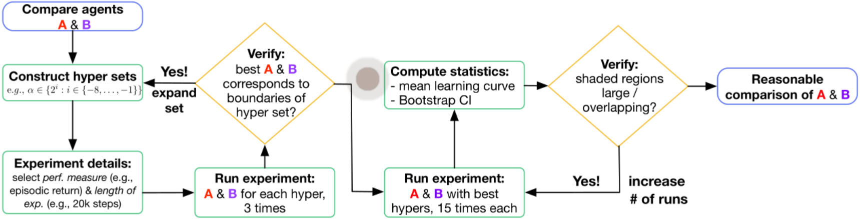


figures and story are from Patterson et al. (2023), which is a good take on best practices in DRL

7.3 A word of caution

- Replicating SAC results (Haarnoja et al., 2018a)
- Found undocumented feature
- Cherry-picking seeds (**don't do this!**)
- Patterson et al. (2023) recommends:





figures and story are from Patterson et al. (2023), which is a good take on best practices in DRL

7.3 Summary

- Maximum entropy RL yields a softmax solution
- SQL implements this with KL-divergence
- SAC improves by branching off TD3 with reparametrization
- SAC also uses action squashing and entropy adjustment
- Empirical comparison inconclusive, TD3 and SAC both SOTA

Learning Objectives

LO7.6: Explain maximum entropy RL, SQL and SAC
LO7.7: Explain action squashing and automatic entropy adjustment
LO7.8: Discuss commonalities and differences between policy gradients

7.3 Next lecture

- Next lecture: **exploration**!
- Start with assignment sheet 3
- Questions? Ask them here: `answers.ewi.tudelft.nl`

SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

image source: `xkcd.com`

References I

Kamil Ciosek and Shimon Whiteson. Expected policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2868–2875, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16116.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems, 2021. URL https://arxiv.org/abs/2103.06257.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of Machine Learning Research (ICML)*, volume 80, pages 1587–1596, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of Machine Learning Research (ICML)*, volume 80, pages 1861–1870, 2018a. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL http://arxiv.org/abs/1812.05905.

Sergej Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL https://arxiv.org/abs/1805.00909.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016. URL http://arxiv.org/abs/1509.02971.

Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization - an empirical study on continuous control. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/1910.09191.

References II

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022. ISSN 2504-4990. URL https://doi.org/10.3390/make4010013.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 13. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2000/hash/e8dfff4676a47048d6f0c4ef899593dd-Abstract.html.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning, 2023. URL https://arxiv.org/abs/2304.01315.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of Machine Learning Research (ICML)*, volume 32, pages 387–395, 2014. URL http://proceedings.mlr.press/v32/silver14.html.

Richard S. Sutton, A. Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 17(1):2603–2631, jan 2016. ISSN 1532-4435. URL https://arxiv.org/abs/1503.04269.

Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, USA, 2010. URL https://www.cs.cmu.edu/~bziebart/publications/thesis-bziebart.pdf.