Reinforcement Learning

Deep RL

Alberto Maria Metelli

15th February 2024

Book References

Richard S. Sutton, Andrew G. Barto
Reinforcement Learning: An Introduction (second edition)
Chapter 16

Outline

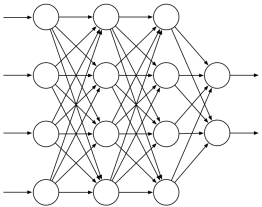
Outline

Deep Value Function Approximation

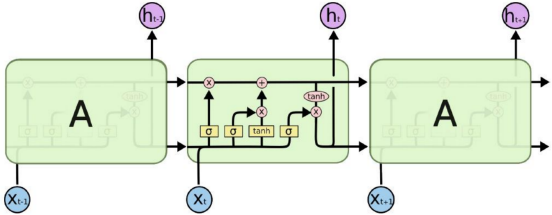
- In previous lectures, we considered features $\mathbf{x}(s)$ that are **fixed**
- We can **learn** the features $\mathbf{x}(s)$ too
- A popular choice is to use **deep neural networks** (DNN) to parametrize $\mathbf{x}(s)$
  - DNN are able to discover **non-linear** features representations specific for a task
  - We can exploit the wide research on architectures and training of DNNs
- Directly parametrize $q_{\mathbf{w}}(s, a)$ as a DNN

Possible Architectures

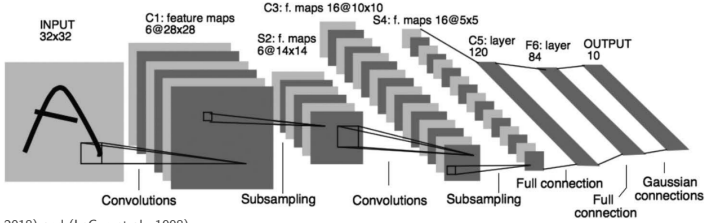
- Multi-Layer Perceptron (MLP)



- Long-Short Term Memory (LSTM)



- Convolutional Neural Networks (CNN)



Pictures from (Sutton and Barto, 2018) and (LeCun et al., 1998)

Deep Q-Learning

- Use a DNN $q_{\mathbf{w}}(s, a)$ to approximate $q_*(s, a)$
- Weight update via **semi-gradient Q-learning**

$$\Delta \mathbf{w} = \alpha \left(R_{t+1} + \gamma \max_{a \in \mathcal{A}} q_{\mathbf{w}}(S_{t+1}, a) - q_{\mathbf{w}}(S_t, A_t) \right) \nabla_{\mathbf{w}} q_{\mathbf{w}}(S_t, A_t)$$

- This is often **implemented** as the minimization of a "loss"

$$L(\mathbf{w}) = \frac{1}{2} \mathbb{E}_{S,A,S' \sim d_\pi} \left[\left(R + \gamma \left[\!\!\left[\max_{a \in \mathcal{A}} q_{\mathbf{w}}(S', a) \right]\!\!\right] - q_{\mathbf{w}}(S, A) \right)^2 \right]$$

  where $[\![\cdot]\!]$ denotes the **stop-gradient**, so that the semi-gradient is $\propto \Delta \mathbf{w}$
- What about the **deadly triad**?

Experience Replay

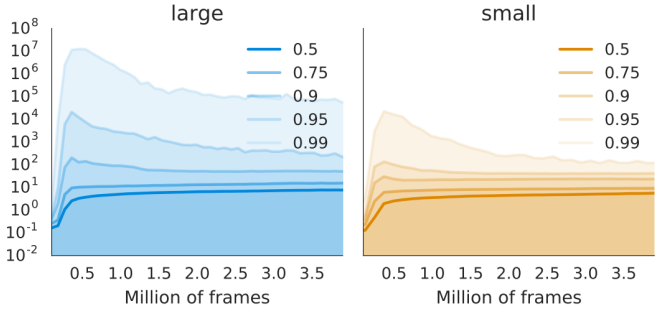
- Consecutive samples are **statistically dependent** (Liu and Zou, 2018)
- Avoid **online** update
- A **replay memory/buffer** $\mathcal{B}$ is employed
  - Collect samples (S, A, S', R) with an ϵ-greedy policy and place them in $\mathcal{B}$
  - Sample **uniformly** a **minibatch** of N_{batch} samples from $\mathcal{B}$ to perform the **update**

$$L(\mathbf{w}) = \frac{1}{2}\mathbb{E}_{S,A,S' \sim \mathcal{B}} \left[\left(R + \gamma \left[\!\!\left[\max_{a \in \mathcal{A}} q_{\mathbf{w}}(S', a) \right]\!\!\right] - q_{\mathbf{w}}(S, A) \right)^2 \right]$$

- The replay memory is managed with **FIFO** strategy

The deadly triad in deep RL

- **Unbounded divergence** is rare in practice with DNNs (van Hasselt et al., 2018)
- Typically the values grow but then recover $\rightarrow$ **soft-divergence**
- Percentiles of the maximal absolute Q-value for two network sizes



Pictures from (van Hasselt et al., 2018)

Target Network

- Updating targets too often might generate **instability** (Mnih et al., 2013)
- Use a **target network** $q_{\mathbf{w}^-}(s, a)$
  - A separate copy of the **online** network $q_{\mathbf{w}}(s, a)$ with weights $\mathbf{w}^-$
  - Updated **periodically** (every few hundred/thousands updates)
  - Used to perform bootstrap

$$L(\mathbf{w}) = \frac{1}{2}\mathbb{E}_{S,A,S' \sim \mathcal{B}}\left[\left(R + \gamma\left[\!\left[\max_{a \in \mathcal{A}} q_{\mathbf{w}^-}(S', a)\right]\!\right] - q_{\mathbf{w}}(S, A)\right)^2\right]$$

- We call this approach **target Q-learning**

DQN Algorithm

Initialize the replay buffer $\mathcal{B}$ with capacity N
Initialize the action-value function $q_{\mathbf{w}}(s, a)$ with random weights $\mathbf{w}$
Initialize the action-value function $q_{\mathbf{w}^-}(s, a)$ with weights $\mathbf{w}^- = \mathbf{w}$
loop for every episode
  Set the initial state S_0
  for $t = 0, \ldots, T-1$ **do**
    With probability ϵ select a random action A_t, otherwise select $A_t \in \arg\max_{a \in \mathcal{A}} q_{\mathbf{w}}(S_t, a)$
    Execute action A_t and observe the reward R_{t+1} and the next state S_{t+1}
    Store transition $(S_t, A_t, R_{t+1}, S_{t+1})$ in the replay buffer $\mathcal{B}$

    Sample a random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from the replay buffer $\mathcal{B}$
    Set $y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma \max_{a' \in \mathcal{A}} q_{\mathbf{w}^-}(S_{j+1}, a') & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - q_{\mathbf{w}}(S_j, A_j))^2$ w.r.t. the network parameters $\mathbf{w}$
    Every C steps, reset the target network $\mathbf{w}^- = \mathbf{w}$
  end for
end loop

Outline

Double (Deep) Q-learning

- Q-learning suffers from **overestimation bias**
  - Even if $q_{\mathbf{w}}(s,a)$ is unbiased estimator for $q_*(s,a)$ (van Hasselt, 2010; van Hasselt et al., 2016)

$$\mathbb{E}\left[\max_{a\in\mathcal{A}} q_{\mathbf{w}}(s,a)\right] \geq \max_{a\in\mathcal{A}} \mathbb{E}[q_{\mathbf{w}}(s,a)] = v_*(s)$$

- **Double Q-Learning**
  - Keep two (ideally) **independent** estimates $q_{\mathbf{w}^A}(s,a)$ and $q_{\mathbf{w}^B}(s,a)$
  - Update $q_{\mathbf{w}^A}$ using $q_{\mathbf{w}^B}$ for the **target**, but evaluated in the greedy action w.r.t. $q_{\mathbf{w}^A}$ and vice versa

$$L^A(\mathbf{w}^A) = \frac{1}{2}\mathbb{E}_{S,A,S'\sim d_\pi}\left[\left(R + \gamma\left[\!\left[q_{\mathbf{w}^B}(S', \arg\max_{a'\in\mathcal{A}} q_{\mathbf{w}^A}(S',a'))\right]\!\right] - q_{\mathbf{w}^A}(S,A)\right)^2\right]$$

Double (Deep) Q-learning

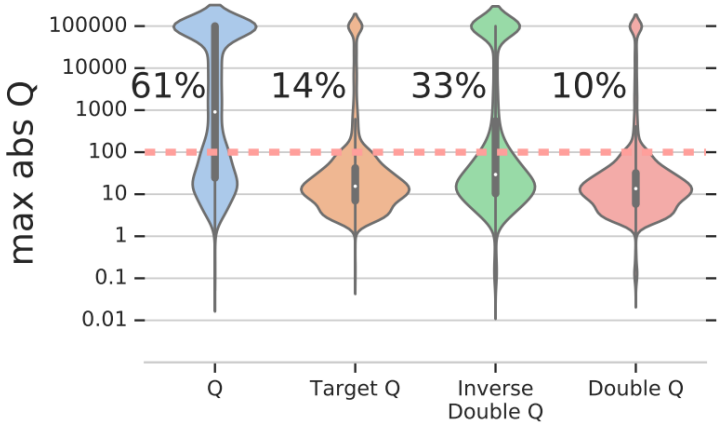
- Alternative version **Inverse Double Q-learning**, less used

$$L^A(\mathbf{w}^A) = \frac{1}{2}\mathbb{E}_{S,A,S' \sim d_\pi}\left[\left(R + \gamma\left[\!\left[q_{\mathbf{w}^A}(S', \arg\max_{a' \in \mathcal{A}} q_{\mathbf{w}^B}(S', a'))\right]\!\right] - q_{\mathbf{w}^A}(S, A)\right)^2\right]$$

- Direct combination with **target network**: $q_{\mathbf{w}^B}$ is the target network
- **Double DQN** (van Hasselt et al., 2018)

$$L(\mathbf{w}) = \frac{1}{2}\mathbb{E}_{S,A,S' \sim \mathcal{B}}\left[\left(R + \gamma\left[\!\left[q_{\mathbf{w}^-}(S', \arg\max_{a' \in \mathcal{A}} q_{\mathbf{w}}(S', a'))\right]\!\right] - q_{\mathbf{w}}(S, A)\right)^2\right]$$

Effect on the Deadly Triad



Pictures from (van Hasselt et al., 2018)

Outline

Prioritized Experience Replay

- DQN samples **uniformly** from the replay buffer $\mathcal{B}$ (Schaul et al., 2015)
- **Prioritize** transitions that are "more interesting"
- One possibility: the priority of sample (S_k, A_k, R_k, S'_k)

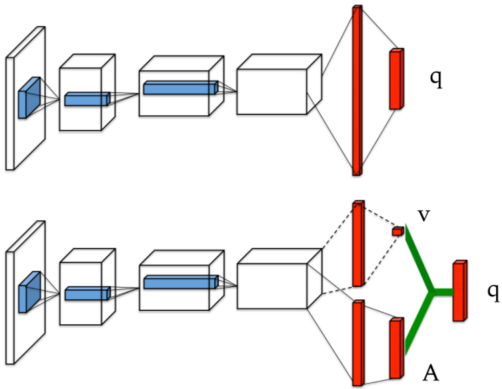
$$p_k \propto |\delta_k|^\alpha \qquad \delta_k = R_k + \gamma q_{\mathbf{w}^-}(S'_k, \arg\max_{a' \in \mathcal{A}} q_{\mathbf{w}}(S'_k, a')) - q_{\mathbf{w}}(S_k, A_k)$$

- Sample according to distribution p_k

Outline

Dueling Networks

- Separate the estimation of the **value function** $v_{\mathbf{w}_1}(s)$ from the **advantage function** $A_{\mathbf{w}_2}(s,a)$ (Wang et al., 2016)
- Compute the **Q-function** as $q_{\mathbf{w}}(s,a) = v_{\mathbf{w}_1}(s) + A_{\mathbf{w}_2}(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a' \in \mathcal{A}} A_{\mathbf{w}_2}(s,a')$, where $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2)$
- We can learn which states are (or not) **valuable** without having to learn the effect of all actions



Pictures taken from (Wang et al., 2016)

Other Tricks and Variants

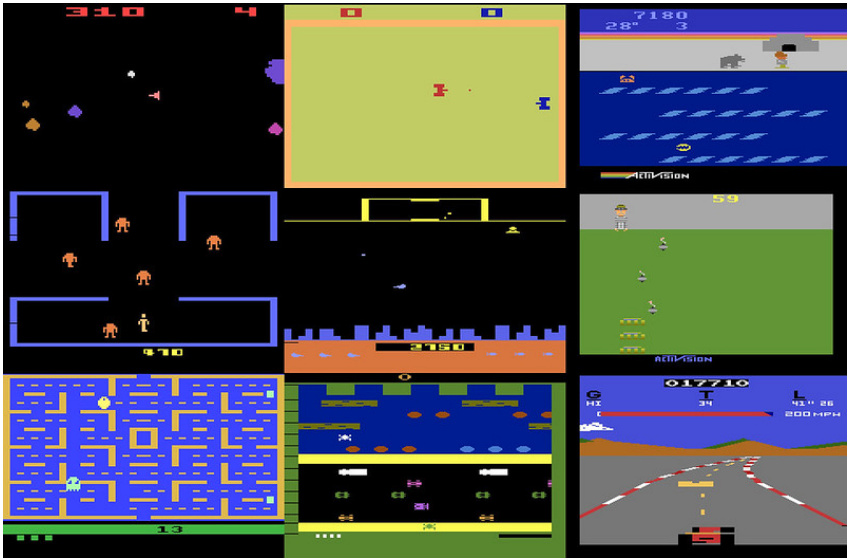
- Multi-step learning (Sutton and Barto, 2018)
  - Use a combination of different n-step targets

$$\sum_{i=0}^{n-1} \lambda_i \left(\sum_{t=0}^{i} \gamma^t R_t + \gamma^{i+1} \max_{a' \in \mathcal{A}} q_{\mathbf{w}}(S_{i+1}, a') \right) \qquad \sum_{i=0}^{n-1} \lambda_i = 1$$

- Distributional DQN (Bellemare et al., 2017)
  - Standard RL learns the **expectation** of the return
  - We learn the full **distribution** of the return
  - Can implement risk-aversion
  - Works well even for expected RL
- Bootstrapped DQN (Osband et al., 2016), Rainbow (Hessel et al., 2018), ...

Outline

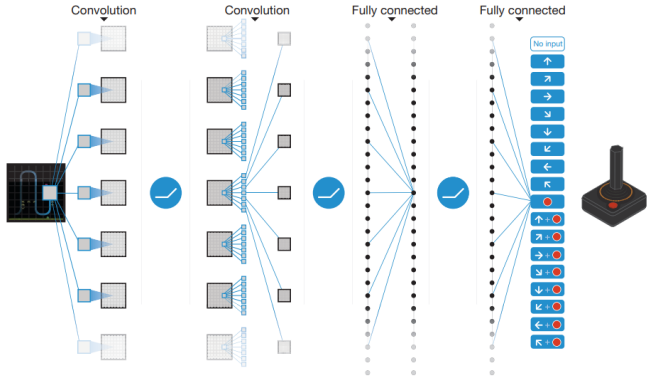
Atari Games



Pictures from https://www.flickr.com/photos/astroguy/2948193385

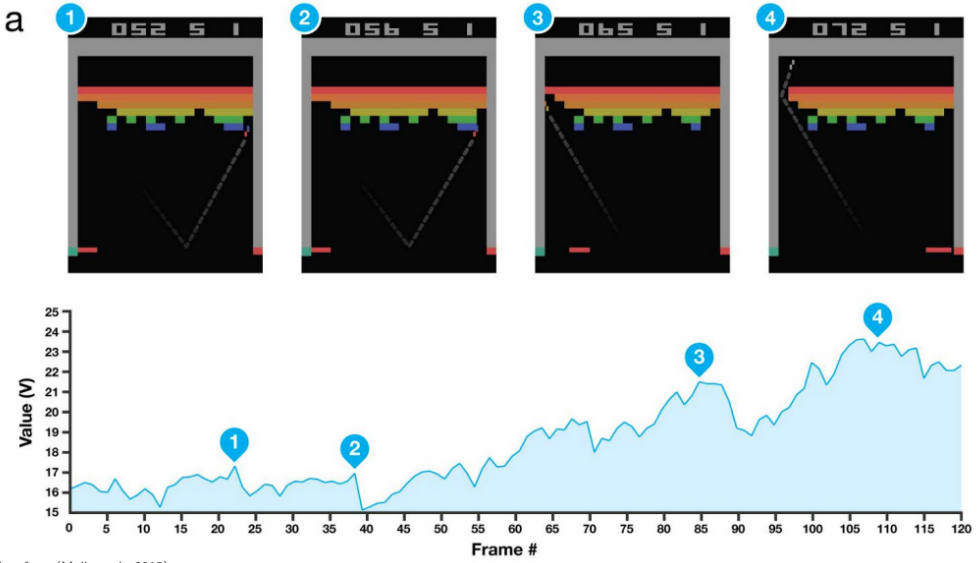
DQN on Atari

- States are stack of raw pixels from last 4 frames 84×84 pixels (Mnih et al., 2015)
- Actions are associated to the 18 joystick/button positions
- Reward is change in score for that step
- Network architecture and hyperparameters fixed across all games, ReLU non-linearity
  - 32 filters 8×8 with stride 4
  - 64 filters 4×4 with stride 2
  - 64 filters 3×3 with stride 1
  - Fully connected 512 neurons



Pictures taken from (Mnih et al., 2015)

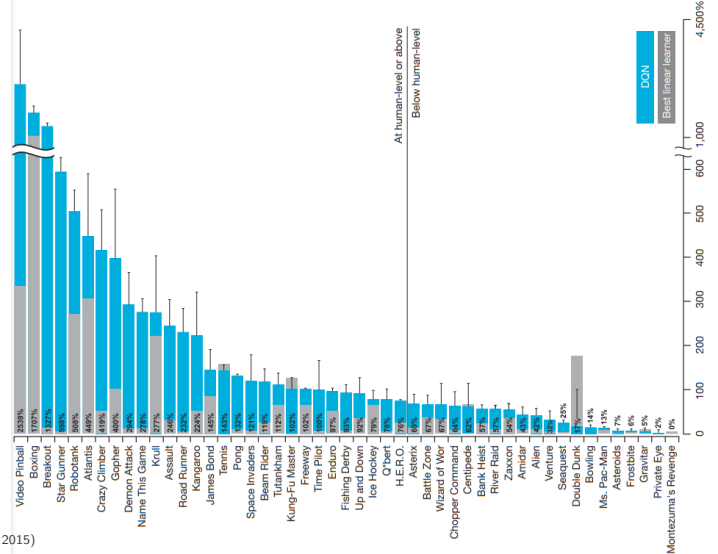
Breakout



Pictures taken from (Mnih et al., 2015)

Results

100 × (DQN score − random play score) / (human score − random play score)



Pictures taken from (Mnih et al., 2015)

Outline

Outline

Sample Reuse

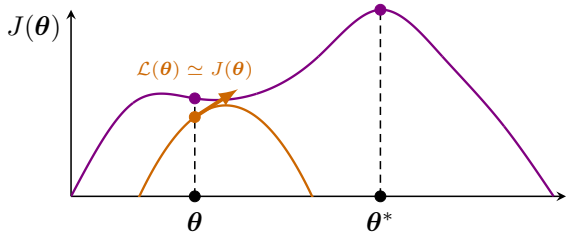
- We would like to **reuse** the samples collected in the past
- We could use **importance weighting** but **variance** might explode
- Use a **surrogate loss** function $\mathcal{L}$ that might be **biased**, but with smaller variance

True loss

$J(\boldsymbol{\theta})$

Surrogate objective

$\mathcal{L}(\boldsymbol{\theta})$



- A surrogate loss is good if

$$\arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} J(\boldsymbol{\theta}) \approx \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$$

Outline

Performance Improvement

- We derive the **surrogate loss** based on the following results
- Introduce the symbol $J(\pi)$ to denote the **expected return** of a policy

$$J(\pi) \coloneqq \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r(S_t, A_t) | S_0 \sim d_0, A_t \sim \pi(\cdot|S_t)\right]$$

Theorem (Performance Difference Lemma (Kakade and Langford, 2002))

Given two policies π and π' we have:

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}} [A_\pi(S, A)]$$

where $d_{\pi'}(s) = \sum_{t=0}^{+\infty} \gamma^t \Pr(S_t = s|\pi', S_0 \sim d_0)$ *and* $A_\pi(s, a) = q_\pi(s, a) - v_\pi(s).$

Proof of the Performance Difference Lemma

Proof.

$$\begin{aligned}
\mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}[A_\pi(S,A)] &= \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}[q_\pi(S,A) - v_\pi(S)] \\
&= \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}[r(S,A)] + \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}\left[\gamma \int_y p(y|S,A) v_\pi(y) \mathrm{d}y - v_\pi(S)\right] \\
&= J(\pi') + \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}\left[\gamma \int_y p(y|S,A) v_\pi(y) \mathrm{d}y\right] - \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}[v_\pi(S)]
\end{aligned}$$

We consider the term in green:

$$\begin{aligned}
\mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}}\left[\gamma \int_y p(y|S,A) v_\pi(y) \mathrm{d}y\right] &= \int_s \left(\sum_{t=0}^{+\infty} \gamma^t \Pr(S_t = s | S_0 \sim d_0, \pi')\right) \gamma \int_a \int_y \pi'(a|s) p(y|s,a) v_\pi(s) \mathrm{d}y \mathrm{d}a \mathrm{d}s \\
&= \int_y \left(\sum_{t=0}^{+\infty} \gamma^t \Pr(S_t = y | S_0 \sim d_0, \pi') - d_0(s)\right) v_\pi(y) \mathrm{d}y \\
&= \mathbb{E}_{S\sim d'_\pi}[v_\pi(S)] - J(\pi)
\end{aligned}$$

□

Surrogate Loss Construction

- We would like to **maximize** $J(\pi')$ over π'
- This is equivalent to maximize the **performance improvement** as π is fixed

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}} [A_\pi(S, A)]$$

- Suppose we have samples from the old policy π, the expectations are w.r.t. to the new policy π'
- We could use **importance weighting** as

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S\sim d_{\pi} \\ A\sim\pi(\cdot|S)}} \left[\frac{d_{\pi'}(S)\pi'(A|S)}{d_\pi(S)\pi(A|S)} A_\pi(S, A) \right]$$

- Still the **variance** problem remains

Surrogate Loss Construction

- Instead, we derive a **lower bound** to the performance improvement

$$J(\pi') - J(\pi) = \underbrace{\mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi'(\cdot|S)}} [A_\pi(S,A)]}_{\text{(A)}} - \underbrace{\int_s (d_\pi(s) - d_{\pi'}(s))\, \mathbb{E}_{A\sim\pi'(\cdot|s)} [A_\pi(s,A)]\, \mathrm{d}s}_{\text{(B)}}$$

- For (A), we use **importance weighting** but on the policy only

$$\text{(A)} = \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}} \left[\frac{\pi'(A|S)}{\pi(A|S)} A_\pi(S,A) \right]$$

- For (B), we bound with the **policy total variation divergence** (Pirotta et al., 2013)

$$\text{(B)} \leq \frac{2\gamma\epsilon}{(1-\gamma)^2} \mathbb{E}_{S\sim d_\pi} \left[D_{\mathsf{TV}} \left(\pi'(\cdot|S), \pi(\cdot|S) \right) \right]$$

where

$$\epsilon := \sup_{s\in\mathcal{S}} \left| \mathbb{E}_{A\sim\pi'(\cdot|s)} [A_\pi(s,A)] \right| \qquad D_{\mathsf{TV}} \left(\pi'(\cdot|s), \pi(\cdot|s) \right) := \frac{1}{2} \int_a |\pi'(\cdot|s) - \pi(\cdot|s)|\, \mathrm{d}a$$

Surrogate Loss

- The resulting **surrogate loss** is given by

$$\mathcal{L}(\pi') \coloneqq \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}}\left[\frac{\pi'(A|S)}{\pi(A|S)}A_\pi(S,A)\right] - \frac{2\gamma\epsilon}{(1-\gamma)^2}\mathbb{E}_{S\sim d_\pi}\left[D_{\mathsf{TV}}\left(\pi'(\cdot|S),\pi(\cdot|S)\right)\right]$$

- For every π', we have $J(\pi) \geq \mathcal{L}(\pi)$
- If $\pi = \pi'$ a.s. then $\mathcal{L}(\pi) = J(\pi)$
- If policy π' is close to π, $\mathcal{L}$ is a good surrogate for J
- **Monotonic Performance Improvement** (Pirotta et al., 2013)

$$\text{If} \qquad \pi^+ \in \arg\max_{\pi'} \mathcal{L}(\pi') \qquad \text{then} \qquad J(\pi^+) \geq J(\pi)$$

- Several approaches proposed based on this surrogate objective (Kakade and Langford, 2002; Wagner, 2011; Pirotta et al., 2013; Schulman et al., 2015a; Achiam et al., 2017)

Outline

Towards TRPO

- The direct optimization of $\mathcal{L}(\pi')$ is not trivial
- The **total variation divergence** D_{TV} is hard to optimize (non-differentiable)
  - We bound it with the **KL-divergence** via Pinsker's inequality

$$D_{\text{TV}}\left(\pi'(\cdot|s), \pi(\cdot|s)\right) \leq \sqrt{\frac{1}{2} D_{\text{KL}}\left(\pi'(\cdot|s), \pi(\cdot|s)\right)}$$

- The optimization of the penalized objective is too **conservative**
  - We replace the penalization with a **constraint**

$$\max_{\pi'} \mathcal{L}^{\text{TRPO}}(\pi') \coloneqq \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}} \left[\frac{\pi'(A|S)}{\pi(A|S)} A_\pi(S,A)\right]$$
$$\text{s.t. } \mathbb{E}_{S\sim d_\pi}\left[D_{\text{KL}}\left(\pi'(\cdot|S), \pi(\cdot|S)\right)\right] \leq \delta$$

where δ is an additional hyperparameter

Trust Region Policy Optimization

- Policies are parametric $\pi_{\boldsymbol{\theta}}(a|s)$ (Schulman et al., 2015a)
- The advantage function $A_\pi(s,a)$ is estimated using a **critic** $v_{\mathbf{w}}(s)$
  - Usually with **generalized advantage estimation** (Schulman et al., 2015b)
- The **constrained** optimization problem is not really solved as is
  - As we have seen in previous lectures, we use the **second-order approximation**

$$D_{\mathsf{KL}}\left(\pi_{\boldsymbol{\theta}'}(\cdot|S), \pi_{\boldsymbol{\theta}}(\cdot|S)\right) \approx \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathrm{T}}\mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

  where $\mathbf{F}(\boldsymbol{\theta})$ is the Fisher Information Matrix
  - Also the objective $\mathcal{L}^{\mathsf{TRPO}}(\boldsymbol{\theta}')$ is approximated at the first order (i.e., using the **gradient**)
- The resulting problem becomes, setting $\mathbf{g} := \nabla_{\boldsymbol{\theta}'}\mathcal{L}^{\mathsf{TRPO}}(\boldsymbol{\theta}')|_{\boldsymbol{\theta}'=\boldsymbol{\theta}}$

$$\max_{\boldsymbol{\theta}'} \mathbf{g}^{\mathrm{T}}(\boldsymbol{\theta}' - \boldsymbol{\theta})$$
$$\text{s.t. } \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathrm{T}}\mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) \leq \delta$$

Trust Region Policy Optimization

- This is a **linear** problem with **convex (quadratic)** constraint $\rightarrow$ **closed-form** solution with Lagrangian duality

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \underbrace{\sqrt{\frac{2\delta}{\|\mathbf{g}\|^2_{\mathbf{F}(\boldsymbol{\theta})^{-1}}}}}_{\text{stepsize}} \cdot \underbrace{\mathbf{F}(\boldsymbol{\theta})^{-1}\mathbf{g}}_{\text{natural gradient}}$$

- Since this is an approximation of the original objective, we cannot "trust" the KL-constraint $\rightarrow$ introduce an additional stepsize
  - **Line search** to ensure performance improvement and enforce the δ constraint
- Inverting the matrix $\mathbf{F}(\boldsymbol{\theta})$ is expensive
  - Use **conjugate gradient** to solve the system $\mathbf{F}(\boldsymbol{\theta})\mathbf{x} = \mathbf{g}$
  - Just requires being able to compute the matrix-vector product $\mathbf{F}(\boldsymbol{\theta})\mathbf{x}$:

$$\mathbf{F}(\boldsymbol{\theta})\mathbf{x} = \mathcal{H}_{\boldsymbol{\theta}} D_{\text{KL}}(\boldsymbol{\theta}' \| \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left(\left(\nabla_{\boldsymbol{\theta}} D_{\text{KL}}(\boldsymbol{\theta}' \| \boldsymbol{\theta}) \right)^{\text{T}} \mathbf{x} \right)$$

TRPO Algorithm

Initialize the actor parameters $\boldsymbol{\theta}$ randomly
Initialize the critic parameters $\mathbf{w}$ randomly
loop for every iteration
Collect a set of trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ by running $\pi_{\boldsymbol{\theta}}$ in the environment
Compute the advantage function $A_{\mathbf{w}}$ (using any method of advantage estimation) based on the current critic $v_{\mathbf{w}}$
Estimate the loss gradient as

$$\mathbf{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t^i | S_t^i) A_{\mathbf{w}}(S_t^i, A_t^i)$$

Use the conjugate gradient to estimate natural gradient $\mathbf{x} = \mathbf{F}(\boldsymbol{\theta})^{-1}\mathbf{g}$
Update the actor parameters $\boldsymbol{\theta}$ via line search

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \sqrt{\frac{2\delta}{\mathbf{x}^{\mathrm{T}}\mathbf{F}(\boldsymbol{\theta})\mathbf{x}}}\mathbf{x}$$

where α is such that it improves the sample loss and satisfies the sample KL-divergence constraint
Fit the critic parameters $\mathbf{w}$ via regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(v_{\mathbf{w}}(S_t^i) - G(\tau_{i,t:T}) \right)^2$$

typically via some gradient descent algorithm
end loop

Outline

From TRPO to PPO

- More **practical** version of TRPO (Schulman et al., 2017)
- Avoid computing the **natural gradient**
- Approximation the **KL-divergence constraint**
- **First Possibility**: replace the constraint with and **adaptive penalization**

$$\boldsymbol{\theta}_{k+1} \in \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}} \left[\frac{\pi_{\boldsymbol{\theta}'}(A|S)}{\pi_{\boldsymbol{\theta}}(A|S)} A_\pi(S,A)\right] - \lambda_k \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}} \left[D_{\mathsf{KL}}\left(\pi_{\boldsymbol{\theta}'}(\cdot|S), \pi_{\boldsymbol{\theta}}(\cdot|S)\right)\right]$$

- Heuristic λ_k adaptation

$$\lambda_{k+1} = \begin{cases} 2\lambda_k & \text{if } \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}} \left[D_{\mathsf{KL}}\left(\pi_{\boldsymbol{\theta}'}(\cdot|S), \pi_{\boldsymbol{\theta}}(\cdot|S)\right)\right] \geq 1.5\delta \\ \frac{\lambda_k}{2} & \text{if } \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}} \left[D_{\mathsf{KL}}\left(\pi_{\boldsymbol{\theta}'}(\cdot|S), \pi_{\boldsymbol{\theta}}(\cdot|S)\right)\right] \leq \frac{\delta}{1.5} \\ \lambda_k & \text{otherwise} \end{cases}$$

From TRPO to PPO

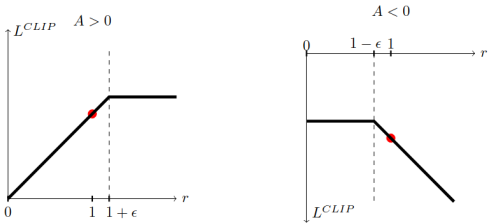
- **Second Possibility**: **clipped** objective function
  - Original TRPO **surrogate loss** function

$$\mathcal{L}^{\text{TRPO}}(\boldsymbol{\theta}') = \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\frac{\pi_{\boldsymbol{\theta}'}(A|S)}{\pi_{\boldsymbol{\theta}}(A|S)}A_{\pi}(S,A)\right] = \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A)A_{\pi}(S,A)\right]$$

  - Construct a lower bound by clipping

$$\mathcal{L}^{\text{PPO}}(\boldsymbol{\theta}') \coloneqq \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\min\left\{\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A)A_{\pi}(S,A), \text{clip}(\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A), 1-\epsilon, 1+\epsilon)A_{\pi}(S,A)\right\}\right]$$

- Clipping empirically prevents policy from moving too much away from $\boldsymbol{\theta}$



Pictures from (Schulman et al., 2017)

PPO Algorithm with Clipping

Initialize the actor parameters $\boldsymbol{\theta}$ randomly
Initialize the critic parameters $\mathbf{w}$ randomly
loop for every iteration
Collect a set of trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ by running $\pi_{\boldsymbol{\theta}}$ in the environment
Compute the advantage function $A_{\mathbf{w}}$ (using any method of advantage estimation) based on the current critic $v_{\mathbf{w}}$
Update the actor parameters $\boldsymbol{\theta}$ maximizing the PPO objective

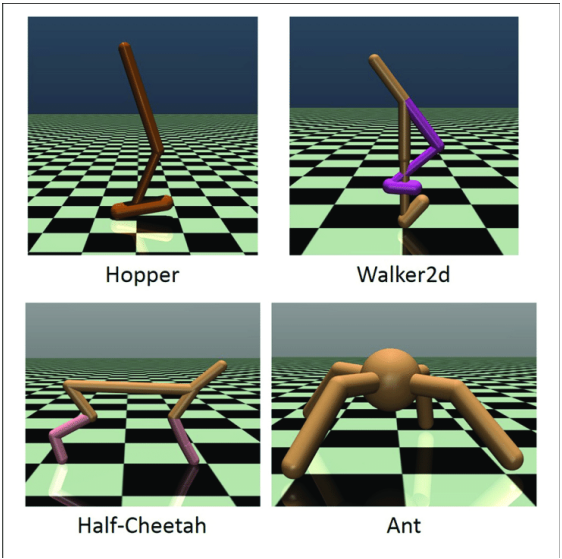
$$\max_{\boldsymbol{\theta}' \in \mathbb{R}^d} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \min \left\{ \omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S_t^i, A_t^i) A_{\mathbf{w}}(S_t^i, A_t^i), \text{clip}(\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S_t^i, A_t^i), 1-\epsilon, 1+\epsilon) A_{\mathbf{w}}(S_t^i, A_t^i) \right\}$$

typically via gradient ascent with Adam
Fit the critic parameters $\mathbf{w}$ via regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(v_{\mathbf{w}}(S_t^i) - G(\tau_{i,t:T}) \right)^2$$

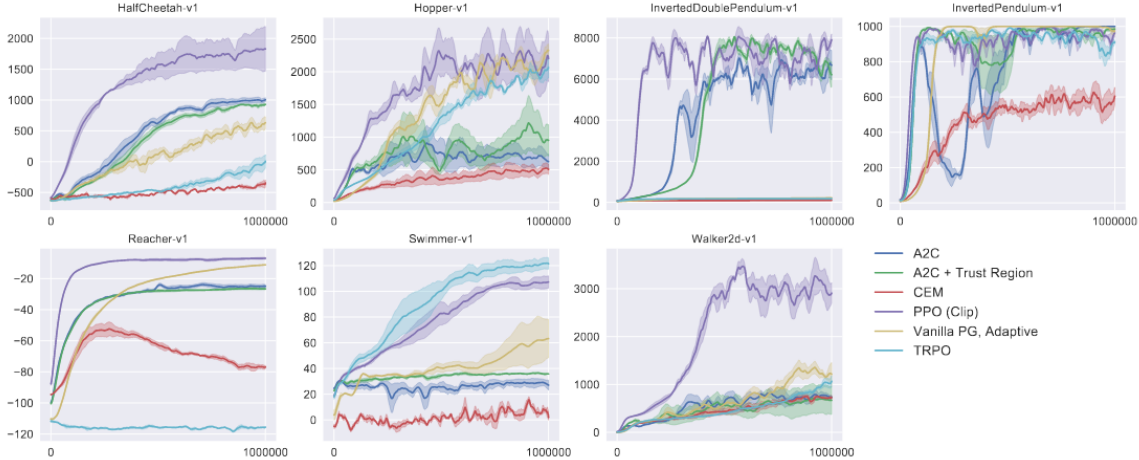
typically via some gradient descent algorithm
end loop

Mujoco Environments



Pictures from (Uchibe, 2018)

PPO - Experimental Results



Pictures from (Schulman et al., 2017)

Outline

Outline

Policy Gradient and Deterministic Policies

- Standard policy gradient requires **stochastic policies** $\pi_{\boldsymbol{\theta}}(a|s)$ for the gradient computation

$$\nabla_{\boldsymbol{\theta}} \int_a \pi_{\boldsymbol{\theta}}(a|s) p(s'|s,a) \mathrm{d}a = \int_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) p(s'|s,a) \mathrm{d}a = \int_a \pi(a|s) p(s'|s,a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \mathrm{d}a$$

- Can we do the same for **deterministic policies** $\mu_{\boldsymbol{\theta}}(s)$?
- We need to differentiate the transition model (Silver et al., 2014)

$$\nabla_{\boldsymbol{\theta}} p(s'|s, \mu_{\boldsymbol{\theta}}(s)) = \nabla_a p(s'|s,a)|_{a=\mu_{\boldsymbol{\theta}}(s)} \nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)$$

Deterministic Policy Gradient Theorem

- Policy Gradient Theorem can be used with **stochastic policies** only (Sutton et al., 1999)

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) q_{\pi_{\boldsymbol{\theta}}}(S,A)]$$

- Extended for **deterministic policies** $\mu_{\boldsymbol{\theta}}(s)$, in (Silver et al., 2014)

Theorem (Deterministic Policy Gradient Theorem (Silver et al., 2014))

Let $\mu_{\boldsymbol{\theta}}(s)$ be a **deterministic** *policy* **differentiable** *in $\boldsymbol{\theta}$ and suppose that the Q-function is* **differentiable in the action** *argument. Then, it holds that:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{S\sim d_{\mu_{\boldsymbol{\theta}}}} \left[\nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(S) \nabla_a q_{\mu_{\boldsymbol{\theta}}}(S,a)|_{a=\mu_{\boldsymbol{\theta}}(S)}\right]$$

- Sufficient condition for making $q_{\mu_{\boldsymbol{\theta}}}(s,a)$ differentiable in a (Silver et al., 2014):

$$p(s'|s,a),\ \nabla_a p(s'|s,a),\ \mu_{\boldsymbol{\theta}}(s),\ \nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(s),\ r(s,a),\ \nabla_a r(s,a),\ d_0(s) \quad \text{are continuous in } s, a, s', \boldsymbol{\theta}$$

Deterministic Policy Gradient

- To compute the **deterministic policy gradient** we need to estimate the Q-function with $q_{\mathbf{w}}(s, a)$
  - The function approximator $q_{\mathbf{w}}(s, a)$ must be **differentiable** w.r.t. a
- **Exploration** problem: explicitly force exploration at the level of actions
  - Use a **noisy policy** for exploration

$$A_t = \mu_{\boldsymbol{\theta}}(S_t) + \epsilon$$

where ϵ is a noise sampled from a suitable distribution
  - **Deterministic Policy Gradient (DPG)** (Silver et al., 2014) is inherently **off-policy**
- **Compatible** value function approximation is extended to DPG (Silver et al., 2014)

Outline

(Deep) Deterministic Policy Gradient

- **Deep Deterministic Policy Gradient (DDPG)**: extension to deep policy and value approximators (Lillicrap et al., 2016)
  - Replay buffer $\mathcal{B}$ to brake dependences
  - Target network for both policy $\mu_{\boldsymbol{\theta}^-}(s)$ and value function $q_{\mathbf{w}^-}(s,a)$
    - No copy every C steps
    - **Polyak** averaging every step

$$\boldsymbol{\theta}^- \leftarrow \rho\boldsymbol{\theta}^- + (1-\rho)\boldsymbol{\theta}$$
$$\mathbf{w}^- \leftarrow \rho\mathbf{w}^- + (1-\rho)\mathbf{w}$$

    where $\rho \in (0,1]$ is a hyperparameter
  - Policy $\mu_{\boldsymbol{\theta}}(s)$ trained to maximize the expected returns

$$\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{S\sim\mathcal{B}}\left[q_{\mathbf{w}}(S, \mu_{\boldsymbol{\theta}}(S))\right]$$

  - Exploration: $\epsilon \sim \mathcal{N}$, where $\mathcal{N}$ is an Ornstein-Uhlenbec process

DDPG Algorithm

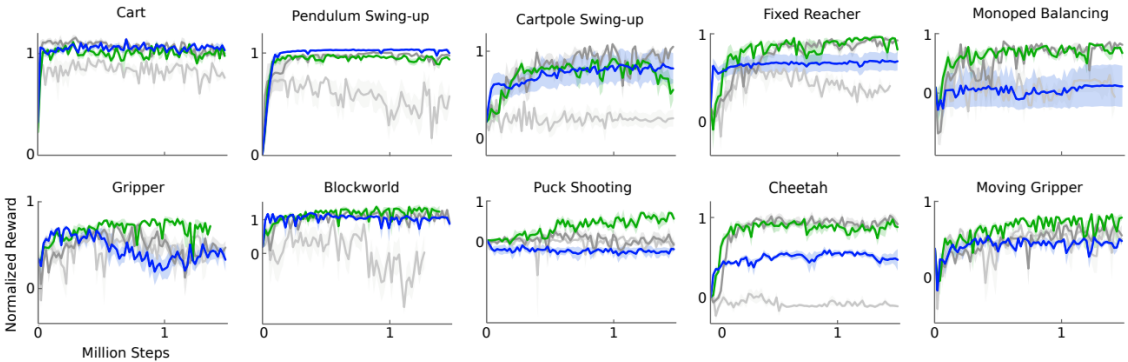
Initialize the replay buffer $\mathcal{B}$ with capacity N
Initialize the actor parameters $\boldsymbol{\theta}$ randomly
Initialize the critic parameters $\mathbf{w}$ randomly
Initialize the target actor parameters $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
Initialize the target critic parameters $\mathbf{w}^- \leftarrow \mathbf{w}$
loop for every iteration
Observe the state S and select action $A = \mu_{\boldsymbol{\theta}}(S) + \epsilon$ where $\epsilon \sim \mathcal{N}$
Execute A and observe the next state S' and reward R
Store (S, A, R, S') in the replay buffer $\mathcal{B}$
Sample a random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from the replay buffer $\mathcal{B}$

Compute the targets

$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma q_{\mathbf{w}^-}(S_{j+1}, \mu_{\boldsymbol{\theta}^-}(S_{j+1})) & \text{otherwise} \end{cases}$$

Perform a gradient descent step on $(y_j - q_{\mathbf{w}}(S_j, A_j))^2$ w.r.t. the critic parameters $\mathbf{w}$
Perform a gradient ascent step on $q_{\mathbf{w}}(S_j, \mu_{\boldsymbol{\theta}}(S_j))$ w.r.t. the actor parameters $\boldsymbol{\theta}$
Update target networks $\boldsymbol{\theta}^- \leftarrow \rho\boldsymbol{\theta}^- + (1-\rho)\boldsymbol{\theta}$ and $\mathbf{w}^- \leftarrow \rho\mathbf{w}^- + (1-\rho)\mathbf{w}$
end loop

DDPG - Experimental Results



original DPG algorithm (minibatch NFQCA) with batch normalization (light grey)
with target network (dark grey)
with target networks and batch normalization (green)
with target networks from pixel-only inputs (blue)

Pictures from (Lillicrap et al., 2016)

Outline

Outline

Deterministic vs Stochastic Policies

- Standard RL focuses on **deterministic** policies
  - Optimal deterministic policy always **exists**
  - Smaller search space
  - In applications, determinism is appreciated
- **Stochastic** policies have some advantages
  - Larger but continuous search space
  - Stochastic policies less prone to overfitting
  - Implicitly embed **exploration**
  - Escape from local optima

Entropy-Regularized RL

- **Entropy-regularized (or soft)** reward function (Nachum et al., 2017; Haarnoja et al., 2018)

$$r(s,a) + \lambda H(\pi(\cdot|s))$$

  where $\lambda > 0$ is the regularization parameter
- $H(\pi(\cdot|s))$ is the **entropy**

$$H(\pi(\cdot|s)) = -\int_a \pi(a|s) \log \pi(a|s) \mathrm{d}a$$

- The (differential Shannon) **entropy** is an index of uncertainty
  - Large $H(\pi(\cdot|s)) \rightarrow \pi(\cdot|s)$ "very stochastic"
  - Small $H(\pi(\cdot|s)) \rightarrow \pi(\cdot|s)$ "almost deterministic"
- Thus, we are encouraging π to be **stochastic**

Entropy-Regularized RL

- Given a policy π, the **soft value functions** become

$$q_\pi^{\text{soft}}(s,a) \coloneqq r(s,a) + \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^t \Big(r(S_t, A_t) + \lambda H(\pi(\cdot|S_t)) \Big) | S_0 = s, A_0 = a, \pi \right]$$

$$v_\pi^{\text{soft}}(s) \coloneqq \mathbb{E}_{A\sim\pi(\cdot|s)}\left[q_\pi^{\text{soft}}(s,A)\right] + \lambda H(\pi(\cdot|S_t))$$

  - When $\lambda \to 0$ they become the standard value functions

- The **greedy softmax policy** w.r.t. function $Q^{\text{soft}}(s,a)$ is defined as

$$\pi^{\text{soft}}(\cdot|s) = \arg\max_\pi \left\{ \mathbb{E}_{A\sim\pi(\cdot|s)}[q^{\text{soft}}(s,A)] + \lambda H(\pi(\cdot|s)) \right\} = \frac{\exp\left(\frac{q^{\text{soft}}(s,\cdot)}{\lambda}\right)}{\int_a \exp\left(\frac{q^{\text{soft}}(s,a)}{\lambda}\right) \mathrm{d}a}$$

  - Simple derivation as the objective is **concave**
  - When $\lambda \to 0$ it becomes the **greedy** policy

Entropy-Regularized RL

- The value function induced by the greedy softmax policy π^{soft} w.r.t. q^{soft} is given by

$$\begin{aligned} v^{\text{soft}}(s) &= \mathbb{E}_{A\sim\pi^{\text{soft}}(\cdot|s)}[q^{\text{soft}}(s,A)] + \lambda H(\pi^{\text{soft}}(\cdot|s)) \\ &= \lambda \log \int_a \exp\left(\frac{q^{\text{soft}}(s,a)}{\lambda}\right) \mathrm{d}a \end{aligned}$$

- Thus, we can rewrite the greedy softmax policy as

$$\pi^{\text{soft}}(a|s) = \exp\left(\frac{q^{\text{soft}}(s,a) - v^{\text{soft}}(s)}{\lambda}\right)$$

Theorem (Policy Improvement Theorem (Haarnoja et al., 2018))

Let $q_\pi^{soft}(s,a)$ be the soft Q-function of policy π. Let π^{soft} be the greedy softmax policy w.r.t. $q_\pi^{soft}(s,a)$. Then, it holds that:

$$q_{\pi^{soft}}^{soft}(s,a) \geq q_\pi^{soft}(s,a) \qquad \forall (s,a) \in \mathcal{S}\times\mathcal{A}$$

Outline

Soft Actor-Critic

- **Actor-critic** approach in the entropy-regularized setting
  - The **actor** estimates the **greedy softmax policy** $\pi_{\boldsymbol{\theta}}(a|s)$
  - The **critic** estimates the **soft Q-function** $q_{\mathbf{w}}(s,a)$
- The policy update is performed via KL **minimization**

$$\boldsymbol{\theta}' \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{S\sim\mathcal{B}} \left[D_{\mathsf{KL}} \left(\pi_{\boldsymbol{\theta}}(\cdot|S) \middle\| \frac{\exp\left(\frac{q_{\mathbf{w}}(S,\cdot)}{\lambda}\right)}{\int_a \exp\left(\frac{q_{\mathbf{w}}(S,a)}{\lambda}\right) \mathrm{d}a} \right) \right]$$

  - Since $q_{\mathbf{w}}(s,a)$ can be differentiated w.r.t. a, we can use the **reparametrization trick** (Kingma and Welling, 2013)
  - Works with **continuous actions** only

Soft Actor-Critic Algorithm

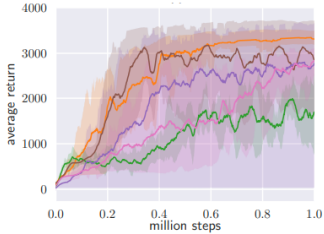
Initialize the replay buffer $\mathcal{B}$ with capacity N
Initialize the actor parameters $\boldsymbol{\theta}$ randomly
Initialize the critic parameters $\mathbf{w}$ randomly
Initialize the target critic parameters $\mathbf{w}^- \leftarrow \mathbf{w}$
loop for every iteration
  Observe the state S and select action $A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$
  Execute A and observe the next state S' and reward R
  Store (S, A, R, S') in the replay buffer $\mathcal{B}$

  Sample a random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from the replay buffer $\mathcal{B}$
  Compute the targets

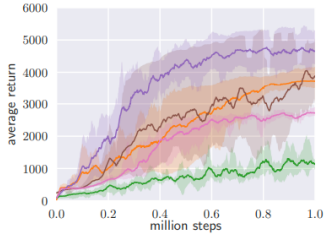
$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma q_{\mathbf{w}^-}(S_{j+1}, A') - \lambda \log \pi_{\boldsymbol{\theta}}(A'|S_{j+1}), \quad A' \sim \pi_{\boldsymbol{\theta}}(\cdot|S_{j+1}) & \text{otherwise} \end{cases}$$

  Perform a gradient descent step on $(y_j - q_{\mathbf{w}}(S_j, A_j))^2$ w.r.t. the critic parameters $\mathbf{w}$
  Perform a gradient descent step on D_{KL} w.r.t. the actor parameters $\boldsymbol{\theta}$ (using reparametrization trick)
  Update the target network $\mathbf{w}^- \leftarrow \rho \mathbf{w}^- + (1-\rho)\mathbf{w}$
end loop

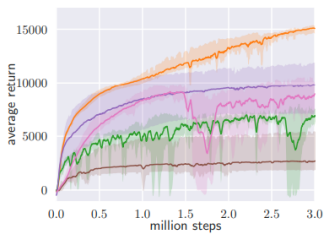
SAC - Experimental Results



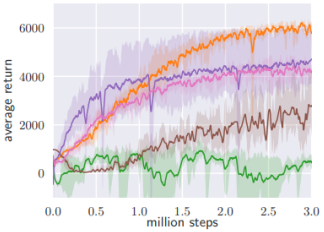
(a) Hopper-v1



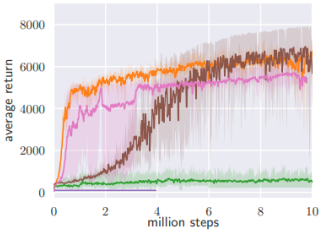
(b) Walker2d-v1



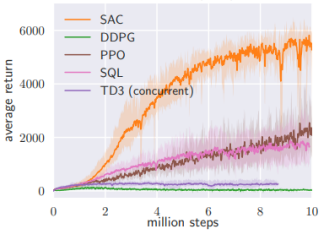
(c) HalfCheetah-v1



(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)

Pictures from (Haarnoja et al., 2018)

Approximation Bound for Entropy-Regularized RL

Exercise 1

Consider the optimal value function v_*:

$$v_*(s) = \max_{\pi \text{ policy}} \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r(S_t, A_t) | S_0 = s\right],$$

and the optimal entropy-regularized value function v_*^{soft}:

$$v_*^{\text{soft}}(s) = \max_{\pi \text{ policy}} \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r(S_t, A_t) + \lambda H(\pi(\cdot|S_t)) | S_0 = s\right]$$

Prove that:

$$\|v_* - v_*^{\text{soft}}\|_\infty \leq \frac{\lambda}{1-\gamma} \max_{s \in \mathcal{S}} H(\pi^*_{\text{soft}}(\cdot|s)) \leq \frac{\lambda}{1-\gamma} |\mathcal{A}| \log |\mathcal{A}|,$$

where π^*_{soft} the optimal policy of the entropy-regularized RL problem.

Outline

Summary

Algorithm	Class	Finite states	Continuous states	Finite actions	Continuous actions	Policy	Reference
DQN	value-based	✓	✓	✓	✗	deterministic	(Mnih et al., 2015)
TRPO	actor-critic	✓	✓	✓ (softmax)	✓ (Gaussian)	stochastic	(Schulman et al., 2015a)
PPO	actor-critic	✓	✓	✓ (softmax)	✓ (Gaussian)	stochastic	(Schulman et al., 2017)
DDPG	actor-critic	✓	✓	✗	✓	deterministic	(Lillicrap et al., 2016)
SAC	actor-critic	✓	✓	✓*	✓	stochastic	(Haarnoja et al., 2018)

* See (Christodoulou, 2019)

What is Missing from the Course?

- Exploration-Exploitation in RL
- Partially observable RL
- Multi-agent RL
- Multi-objective RL
- Non-stationary RL
- Risk-sensitive and robust RL
- Transfer/multi-task/meta RL
- Safe RL
- Hierarchical RL
- Lifelong/continual RL
- Imitation Learning
- Inverse RL
- Online learning and Multi-armed Bandits
- ...

References I

J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2017.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 449–458, 2017. URL `http://proceedings.mlr.press/v70/bellemare17a.html`.

P. Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3215–3222, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204`.

S. M. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 267–274. Morgan Kaufmann, 2002.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

References II

R. Liu and J. Zou. The effects of memory replay in reinforcement learning. In *56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018, Monticello, IL, USA, October 2-5, 2018*, pages 478–485, 2018. doi: 10.1109/ALLERTON.2018.8636075. URL https://doi.org/10.1109/ALLERTON.2018.8636075.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

I. Osband, C. Blundell, A. Pritzel, and B. V. Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4026–4034, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/8d8818c8e140c64c743113f563cf750f-Abstract.html.

M. Pirotta, M. Restelli, A. Pecorino, and D. Calandriello. Safe policy iteration. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 307–315. JMLR.org, 2013.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 37, pages 1889–1897. JMLR.org, 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

References III

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org, 2014.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 1057–1063. The MIT Press, 1999.

E. Uchibe. Cooperative and competitive reinforcement and imitation learning for a mixture of heterogeneous learning modules. *Frontiers in neurorobotics*, page 61, 2018.

H. van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2613–2621, 2010. URL `https://proceedings.neurips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html`.

H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100, 2016. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389`.

H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018. URL `http://arxiv.org/abs/1812.02648`.

P. Wagner. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2573–2581, 2011.

Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, pages 1995–2003, 2016. URL `http://proceedings.mlr.press/v48/wangf16.html`.