Deep Reinforcement Learning and Control

Pathwise derivatives, DDPG, Multigoal RL

Katerina Fragkiadaki
Part of the slides on path wise derivatives adapted from John Schulman
Computing Gradients of Expectations

When the variable w.r.t. which we are differentiating appears in the distribution:

\[ \nabla_{\theta} \mathbb{E}_{x \sim p(\cdot|\theta)} F(x) = \mathbb{E}_{x \sim p(\cdot|\theta)} \nabla_{\theta} \log p(\cdot|\theta) F(x) \]

\[
\text{e.g. } \nabla_{\theta} \mathbb{E}_{a \sim \pi_{\theta}} R(a, s)
\]

likelihood ratio gradient estimator

When the variable w.r.t. which we are differentiating appears in the expectation:

\[ \nabla_{\theta} \mathbb{E}_{z \sim \mathcal{N}(0,1)} F(x(\theta), z) = \mathbb{E}_{z \sim \mathcal{N}(0,1)} \nabla_{\theta} F(x(\theta), z) = \mathbb{E}_{z \sim \mathcal{N}(0,1)} \frac{dF(x(\theta), z)}{dx} \frac{dx}{d\theta} \]

pathwise derivative

Re-parametrisation trick: For some distributions \( p(x|\theta) \) we can switch from one gradient estimator to the other.

Why would we want to do so?
Known MDP

Reward and dynamics are known deterministic node: the value is a deterministic function of its input
stochastic node: the value is sampled based on its input (which parametrizes the distribution to sample from)
deterministic computation node

\[
\begin{align*}
\pi_\theta(s) & \quad r_0 \\
\rho(s, a) & \quad a_0 \\
s_0 & \\
T(s, a) & \\
\pi_\theta(s) & \quad r_1 \\
\rho(s, a) & \quad a_1 \\
s_1 & \\
T(s, a) & \quad \ldots
\end{align*}
\]
I want to learn $\theta$ to maximize the reward obtained.
What if the policy is deterministic?

\[ a = \pi_\theta(s) \]

I want to learn \( \theta \) to maximize the reward obtained.

I can compute the gradient with backpropagation.

\[ \nabla_\theta \rho(s, a) = \rho_a \pi_\theta \theta \]
What if the policy is stochastic?

I want to learn $\theta$ to maximize the reward obtained.

Likelihood ratio estimator, works for both continuous and discrete actions

$$\mathbb{E}_a \nabla_\theta \log \pi_\theta(s, a) \rho(s, a)$$
Policies are parametrized Gaussians

\[ \pi_\theta(s) \]

\[ \rho(s, a) \]

\[ a \sim \mathcal{N}(\mu(s, \theta), \Sigma(s, \theta)) \]

\[ \mu_\theta(s), \sigma_\theta(s) \]

\[ \pi_\theta(s) \]

I want to learn \( \theta \) to maximize the reward obtained.

\[ \mathbb{E}_a \nabla_\theta \log \pi_\theta(s, a) \rho(s, a) \]

If \( \sigma^2 \) is constant:

\[ \nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s; \theta)) \frac{\partial \mu(s; \theta)}{\partial \theta}}{\sigma^2} \]
Re-parametrization for Gaussian

\[ r_0 \]

\[ \rho(s, a) \]

\[ a = \mu(s, \theta) + z \odot \sigma(s, \theta) \]

\[ z \sim \mathcal{N}(0, I) \]

\[ \mu_\theta(s) \]

\[ \sigma_\theta(s) \]

\[ \pi_\theta(s) \]

\[ s_0 \]

\[ \theta \]
Re-parametrization for Gaussian

\[ a = \mu(s, \theta) + z \odot \sigma(s, \theta) \]

\[ \frac{da}{d\theta} = \frac{d\mu(s, \theta)}{d\theta} + z \odot \frac{d\sigma(s, \theta)}{d\theta} \]

\[ \nabla_\theta \mathbb{E}_z \left[ \rho \left( a(\theta, z), s \right) \right] = \mathbb{E}_z - \frac{d\rho \left( a(\theta, z), s \right)}{da} \frac{da(\theta, z)}{d\theta} \]

Sample estimate:

\[ \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \left[ \rho \left( a(\theta, z_i), s \right) \right] = \frac{1}{N} \sum_{i=1}^{N} \frac{d\rho \left( a(\theta, z), s \right)}{da} \frac{da(\theta, z)}{d\theta} \bigg|_{z=z_i} \]
Re-parametrization for Gaussian

\[ r_0 \]

\[ \rho(s, a) \]

\[ \pi_{\theta}(s) \]

\[ s_0 \]

\[ \theta \]

\[ a = \mu(s, \theta) + z \odot \sigma(s, \theta) \]

\[ z \sim \mathcal{N}(0, I) \]

\[ \mu_{\theta}(s), \sigma_{\theta}(s) \]

\[ \mathbb{E}(\mu + z\sigma) = \mu \]

\[ \text{Var}(\mu + z\sigma) = \sigma^2 \]

isotropic

\[ a = \mu(s, \theta) + z \odot \sigma(s, \theta) \]

\[ \frac{da}{d\theta} = \frac{d\mu(s, \theta)}{d\theta} + z \odot \frac{d\sigma(s, \theta)}{d\theta} \]

\[ \nabla_{\theta} \mathbb{E}_z \left[ \rho \left( a(\theta, z), s \right) \right] = \mathbb{E}_z \left[ \frac{d\rho \left( a(\theta, z), s \right)}{da} \frac{da(\theta, z)}{d\theta} \right]_{z=z_i} \]

Sample estimate:

\[ \nabla_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left[ \rho \left( a(\theta, z_i), s \right) \right] = \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{d\rho \left( a(\theta, z), s \right)}{da} \frac{da(\theta, z)}{d\theta} \right]_{z=z_i} \]
Re-parametrization for Gaussian

$\mathbb{E}(\mu + z\sigma) = \mu$

$\text{Var}(\mu + z\sigma) = \sigma^2$

isotropic

$a = \mu(s, \theta) + z \odot \sigma(s, \theta)$

$\frac{da}{d\theta} = \frac{d\mu(s, \theta)}{d\theta} + z \odot \frac{d\sigma(s, \theta)}{d\theta}$

Sample estimate:

$\nabla_{\theta} \mathbb{E}_z \left[ \rho \left( a(\theta, z), s \right) \right] = \mathbb{E}_z \left[ \frac{d\rho \left( a(\theta, z), s \right)}{da} \frac{da(\theta, z)}{d\theta} \right] |_{z=z_i}$

general

$a = \mu(\sigma, \theta) + Lz, \quad \Sigma = LL^T$

The pathwise derivative uses the derivative of the reward w.r.t. the action!
Policies are parametrized Categorical distr

I want to learn $\theta$ to maximize the reward obtained.

$$\max_a \nabla_\theta \log \pi_\theta(s, a) \rho(s, a)$$
Re-parametrization for categorical distributions

Consider variable $y$ following the $K$ categorical distribution:

$$y_k \sim \frac{\exp((\log p_k)/\tau)}{\sum_{j=0}^{K} \exp((\log p_j)/\tau)}$$

Categorical reparametrization with Gumbel-Softmax, Sang et al. 2017
Consider variable $y$ following the $K$ categorical distribution:

$$y_k \sim \frac{\exp((\log p_k)/\tau)}{\sum_{j=0}^{K} \exp((\log p_j)/\tau)}$$

Re-parametrization:

$$a_k = \arg\max_k (\log p_k + \epsilon_k), \quad \epsilon_k = -\log(-\log(U)), \quad u \sim U[0,1]$$
Re-parametrization trick for categorical distributions

Consider variable $y$ following the $K$ categorical distribution:

$$a_k \sim \frac{\exp((\log p_k)/\tau)}{\sum_{j=0}^{K} \exp((\log p_j)/\tau)}$$

Reparametrization:

$$a_k = \arg \max_k (\log p_k + \epsilon_k), \quad \epsilon_k = -\log(-\log(U)), \quad u \sim \mathcal{U}[0,1]$$

In the forward pass you sample from the parametrized distribution

$$a_k \sim G(\log p)$$

In the backward pass you use the soft distribution:

$$\frac{da}{d\theta} = \frac{dG}{dp} \frac{dp}{d\theta}$$

*Categorical reparametrization with Gumbel-Softmax, Sang et al. 2017*
Re-parametrization trick for categorical distributions

Consider variable $y$ following the $K$ categorical distribution:

$$a_k \sim \frac{\exp((\log p_k)/\tau)}{\sum_{j=0}^{K} \exp((\log p_j)/\tau)}$$

Reparametrization:

$$a_k = \arg \max_k (\log p_k + \epsilon_k), \quad \epsilon_k = -\log(-\log(U)), \quad u \sim \mathcal{U}[0,1]$$

In the forward pass you sample from the parametrized distribution

$$a_k \sim G(\log p)$$

In the backward pass you use the soft distribution:

$$\frac{da}{d\theta} = \frac{dG}{dp} \frac{dp}{d\theta}$$

*Categorical reparametrization with Gumbel-Softmax*, Sang et al. 2017
Back-propagating through discrete variables

For binary neurons:

forward pass

backward pass

Straight-through sigmoidal

For general categorically distributed neurons:

forward pass

backward pass

Categorical reparametrization with Gumbel-Softmax, Sang et al. 2017
Episodic MDP:

We want to compute: \( \nabla_\theta \mathbb{E}[R_T] \)
Episodic MDP:

We want to compute: \( \nabla_{\theta} \mathbb{E}[R_T] \)

Reparameterize: \( a_t = \pi(s_t, z_t; \theta) \). \( z_t \) is noise from fixed distribution.
Episodic MDP:

We want to compute: \( \nabla_{\theta} \mathbb{E}[R_T] \)

Reparameterize: \( a_t = \pi(s_t, z_t; \theta) \). \( z_t \) is noise from fixed distribution.

For path wise derivative to work, we need transition dynamics and reward function to be known.
Re-parametrized Policy Gradients

For path wise derivative to work, we need transition dynamics and reward function to be known, or…
Using a Q-function $Q(s_t, a_t)$, we can express the expected return as

\[
\frac{d}{d\theta} \mathbb{E}[R_T] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dR_T}{da_t} \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{da_t} \mathbb{E}[R_T | a_t] \frac{da_t}{d\theta} \right]
\]

\[
= \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t} \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{d\theta} Q(s_t, \pi(s_t, z_t; \theta)) \right]
\]

- Learn $Q_\phi$ to approximate $Q^\pi, \gamma$, and use it to compute gradient estimates.
Learn $Q_\phi$ to approximate $Q^{\pi, \gamma}$, and use it to compute gradient estimates.

Pseudocode:

```plaintext
for iteration=1, 2, ... do
    Execute policy $\pi_\theta$ to collect $T$ timesteps of data
    Update $\pi_\theta$ using $g \propto \nabla_\theta \sum_{t=1}^{T} Q(s_t, \pi(s_t, z_t; \theta))$
    Update $Q_\phi$ using $g \propto \nabla_\phi \sum_{t=1}^{T} (Q_\phi(s_t, a_t) - \hat{Q}_t)^2$, e.g. with TD($\lambda$)
end for
```

What if we give up on stochastic actions?

Deep Deterministic Policy Gradients

\[
\frac{d}{d\theta} \mathbb{E}[R_T] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dR_T}{da_t} \frac{da_t}{d\theta} \right]
\]

Continuous control with deep reinforcement learning, Lilicrap et al. 2016
Deep Deterministic Policy Gradients

This expectation refers to the dynamics after time $t$

$$\frac{d}{d\theta} \mathbb{E} [R_T] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dR_T}{da_t} \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{da_t} \mathbb{E} [R_T | a_t] \frac{da_t}{d\theta} \right]$$

Continuous control with deep reinforcement learning, Lilicarp et al. 2016
Deep Deterministic Policy Gradients

$$\frac{d}{d\theta} \mathbb{E} [R_T] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dR_T}{da_t} \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{da_t} \mathbb{E} [R_T | a_t] \frac{da_t}{d\theta} \right]$$

$$= \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t} \frac{da_t}{d\theta} \right]$$

Continuous control with deep reinforcement learning, Lillicrap et al. 2016
Deep Deterministic Policy Gradients

\[
\frac{d}{d\theta} \mathbb{E}[R_T] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{dR_T}{da_t} \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{da_t} \mathbb{E}[R_T \mid a_t] \frac{da_t}{d\theta} \right] \\
= \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{d\theta} Q(s_t, a_t) \frac{da_t}{d\theta} \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \frac{d}{d\theta} Q(s_t, \pi(s_t; \theta)) \right]
\]

Continuous control with deep reinforcement learning, Lillicrap et al. 2016
Deep Deterministic Policy Gradients

**Algorithm 1 DDPG algorithm**

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^Q \leftarrow \theta^Q$, $\theta^\mu \leftarrow \theta^\mu$
Initialize replay buffer $R$

for episode = 1, $M$ do

Initialize a random process $N$ for action exploration
Receive initial observation state $s_1$

for $t = 1, T$ do

Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^\mu)|\theta^Q')$
Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s_i, a_i|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^Q' \leftarrow \tau \theta^Q + (1 - \tau) \theta^Q'$$
$$\theta^\mu' \leftarrow \tau \theta^\mu + (1 - \tau) \theta^\mu'$$

end for

end for
Deep Deterministic Policy Gradients

\[ a = \mu(\theta) \]

\[ \nabla_{\theta \mu} J \approx E_{s_t \sim \rho^s} \left[ \nabla_{\theta \mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right] \\
= E_{s_t \sim \rho^s} \left[ \nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta \mu} \mu(s|\theta^\mu)|_{s=s_t} \right] \]

We are following a stochastic behavior policy to collect data. Deep Q learning for contours actions -> DDPG
Stochastic Value Gradients V0

\[ z \sim \mathcal{N}(0, 1) \]

\[ a = \mu(s; \theta) + z\sigma(s; \theta) \]

(Where are the other versions? We will see them in the model based RL lecture)
End-to-end model based RL

Re-parametrization trick for both policies and dynamics

\[
\begin{align*}
    r_0 & \quad r = R(a, s) \\
    a_0 & \quad a = \pi(s; z; \theta) \\
    s_0 & \quad s' = \hat{f}(s, a; \xi; \phi) \\

    r_1 & \quad r = R(a, s) \\
    a_1 & \quad a = \pi(s; z; \theta) \\
    s_1 & \quad s' = \hat{f}(s, a; \xi; \phi) \\
    \cdots & \quad \cdots \quad s_T
\end{align*}
\]

Deep Deterministic Policy Gradients

Figure 1: Example screenshots of a sample of environments we attempt to solve with DDPG. In order from the left: the cartpole swing-up task, a reaching task, a gasp and move task, a puck-hitting task, a monoped balancing task, two locomotion tasks and Torcs (driving simulator). We tackle all tasks using both low-dimensional feature vector and high-dimensional pixel inputs. Detailed descriptions of the environments are provided in the supplementary. Movies of some of the learned policies are available at https://goo.gl/J4PIAz.
Deep Deterministic Policy Gradients

Figure 2: Performance curves for a selection of domains using variants of DPG: 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). Target networks are crucial.

State representation input can be pixels or robotic configuration and target locations

https://www.youtube.com/watch?v=tJBlqkC1wWM&feature=youtu.be
## Model Free Methods - Comparison

<table>
<thead>
<tr>
<th>Task</th>
<th>Random</th>
<th>REINFORCE</th>
<th>TNP</th>
<th>RWR</th>
<th>REPS</th>
<th>TRPO</th>
<th>CEM</th>
<th>CMA-ES</th>
<th>DDPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cart-Pole Balancing</td>
<td>77.1±0.0</td>
<td>4693.7±14.0</td>
<td>3986.4±748.9</td>
<td>4861.5±12.3</td>
<td>565.6±137.6</td>
<td>4869.8±37.6</td>
<td>4815.4±4.8</td>
<td>2440.4±568.3</td>
<td>4634.4±87.8</td>
</tr>
<tr>
<td>Inverted Pendulum*</td>
<td>-153.4±0.2</td>
<td>13.4±18.0</td>
<td>209.7±55.5</td>
<td>84.7±13.8</td>
<td>-113.3±4.6</td>
<td>247.2±76.1</td>
<td>38.2±25.7</td>
<td>-40.1±5.7</td>
<td>40.0±244.6</td>
</tr>
<tr>
<td>Mountain Car</td>
<td>-415.4±1.0</td>
<td>-67.1±1.0</td>
<td>-66.5±4.5</td>
<td>-79.4±1.1</td>
<td>-275.6±166.3</td>
<td>-61.7±0.9</td>
<td>-66.0±2.4</td>
<td>-85.0±7.7</td>
<td>-288.4±170.3</td>
</tr>
<tr>
<td>Acrobot</td>
<td>-1904.5±1.0</td>
<td>-508.1±91.0</td>
<td>-395.8±121.2</td>
<td>-352.7±35.9</td>
<td>-1001.5±10.8</td>
<td>-326.0±24.4</td>
<td>-436.8±14.7</td>
<td>-785.6±13.1</td>
<td>-2236±5.8</td>
</tr>
<tr>
<td>Double Inverted Pendulum*</td>
<td>149.7±0.1</td>
<td>4116.5±65.2</td>
<td>4455.4±37.6</td>
<td>3614.8±368.1</td>
<td>446.7±114.8</td>
<td>4412.4±50.4</td>
<td>2566.2±178.9</td>
<td>1576.1±51.3</td>
<td>2863.4±154.0</td>
</tr>
<tr>
<td>Swimmer*</td>
<td>-1.7±0.1</td>
<td>92.3±0.1</td>
<td>96.0±0.2</td>
<td>60.7±5.5</td>
<td>3.8±3.3</td>
<td>96.0±0.2</td>
<td>68.8±2.4</td>
<td>64.9±1.4</td>
<td>85.8±1.8</td>
</tr>
<tr>
<td>Hopper</td>
<td>8.4±0.0</td>
<td>714.0±29.3</td>
<td>1155.1±57.9</td>
<td>553.2±71.0</td>
<td>86.7±17.6</td>
<td>1183.3±150.0</td>
<td>63.1±7.8</td>
<td>20.3±14.3</td>
<td>267.1±43.5</td>
</tr>
<tr>
<td>2D Walker</td>
<td>-1.7±0.1</td>
<td>506.5±78.8</td>
<td>1382.6±108.2</td>
<td>136.0±15.9</td>
<td>-37.0±38.1</td>
<td>1353.8±85.0</td>
<td>84.5±19.2</td>
<td>77.1±24.3</td>
<td>318.4±181.6</td>
</tr>
<tr>
<td>Half-Cheetah</td>
<td>-90.8±0.3</td>
<td>1183.1±69.2</td>
<td>1729.5±184.6</td>
<td>376.1±28.2</td>
<td>34.5±38.0</td>
<td>1914.0±120.1</td>
<td>330.4±274.8</td>
<td>441.3±107.6</td>
<td>2148.6±702.7</td>
</tr>
<tr>
<td>Ant*</td>
<td>13.4±0.7</td>
<td>548.3±55.5</td>
<td>706.0±127.7</td>
<td>37.6±3.1</td>
<td>39.0±9.8</td>
<td>730.2±61.3</td>
<td>49.2±5.9</td>
<td>17.8±15.5</td>
<td>326.2±20.8</td>
</tr>
<tr>
<td>Simple Humanoid</td>
<td>41.5±0.2</td>
<td>1291.3±34.0</td>
<td>285.0±24.5</td>
<td>93.3±17.4</td>
<td>28.5±4.7</td>
<td>269.7±40.3</td>
<td>60.6±12.9</td>
<td>28.7±3.9</td>
<td>99.4±28.1</td>
</tr>
<tr>
<td>Full Humanoid</td>
<td>13.2±0.1</td>
<td>262.2±10.5</td>
<td>288.4±25.2</td>
<td>46.7±5.6</td>
<td>41.7±6.1</td>
<td>287.0±23.4</td>
<td>36.9±2.9</td>
<td>N/A±N/A</td>
<td>119.0±31.2</td>
</tr>
<tr>
<td>Cart-Pole Balancing (LS)*</td>
<td>77.1±0.0</td>
<td>420.9±265.5</td>
<td>945.1±27.8</td>
<td>68.9±1.5</td>
<td>898.1±22.1</td>
<td>960.2±46.0</td>
<td>227.0±233.0</td>
<td>68.0±1.6</td>
<td></td>
</tr>
<tr>
<td>Inverted Pendulum (LS)</td>
<td>-122.1±0.1</td>
<td>-13.4±3.2</td>
<td>0.7±6.1</td>
<td>-107.4±0.2</td>
<td>-87.2±8.0</td>
<td>4.5±4.1</td>
<td>-81.2±33.2</td>
<td>-62.4±3.4</td>
<td></td>
</tr>
<tr>
<td>Mountain Car (LS)</td>
<td>-83.0±0.0</td>
<td>-81.2±0.6</td>
<td>-65.7±9.0</td>
<td>-81.7±0.1</td>
<td>-82.6±0.4</td>
<td>-64.2±9.5</td>
<td>-68.9±1.3</td>
<td>-73.2±0.6</td>
<td></td>
</tr>
<tr>
<td>Acrobot (LS)*</td>
<td>-393.2±0.0</td>
<td>-128.9±11.6</td>
<td>-84.6±2.9</td>
<td>-235.9±5.3</td>
<td>-379.5±1.4</td>
<td>-83.3±9.9</td>
<td>-149.5±15.3</td>
<td>-159.9±7.5</td>
<td></td>
</tr>
<tr>
<td>Cart-Pole Balancing (NO)*</td>
<td>101.4±0.1</td>
<td>616.0±210.8</td>
<td>916.3±23.0</td>
<td>93.8±1.2</td>
<td>99.6±7.2</td>
<td>606.2±122.2</td>
<td>181.4±32.1</td>
<td>104.4±16.0</td>
<td></td>
</tr>
<tr>
<td>Inverted Pendulum (NO)</td>
<td>-122.2±0.1</td>
<td>6.5±11.1</td>
<td>11.5±0.5</td>
<td>-110.0±4.4</td>
<td>-119.3±4.2</td>
<td>10.4±2.2</td>
<td>-55.6±16.7</td>
<td>-80.3±2.8</td>
<td></td>
</tr>
<tr>
<td>Mountain Car (NO)</td>
<td>-83.0±0.0</td>
<td>-74.7±7.8</td>
<td>-64.5±8.6</td>
<td>-81.7±0.1</td>
<td>-82.9±0.1</td>
<td>-60.2±2.0</td>
<td>-67.4±1.4</td>
<td>-73.5±0.5</td>
<td></td>
</tr>
<tr>
<td>Acrobot (NO)*</td>
<td>-393.5±0.0</td>
<td>-186.7±31.3</td>
<td>-164.5±13.4</td>
<td>-233.1±0.4</td>
<td>-258.5±14.0</td>
<td>-149.6±8.6</td>
<td>-213.4±6.3</td>
<td>-236.6±6.2</td>
<td></td>
</tr>
<tr>
<td>Cart-Pole Balancing (SI)*</td>
<td>76.3±0.1</td>
<td>431.7±274.1</td>
<td>980.5±7.3</td>
<td>69.0±2.8</td>
<td>702.4±196.4</td>
<td>980.3±5.1</td>
<td>746.6±93.2</td>
<td>71.6±2.9</td>
<td></td>
</tr>
<tr>
<td>Inverted Pendulum (SI)</td>
<td>-121.8±0.2</td>
<td>-5.3±5.6</td>
<td>14.8±1.7</td>
<td>-108.7±4.7</td>
<td>-92.8±23.9</td>
<td>14.1±0.9</td>
<td>-51.8±10.6</td>
<td>-63.1±4.8</td>
<td></td>
</tr>
<tr>
<td>Mountain Car (SI)</td>
<td>-82.7±0.0</td>
<td>-63.9±0.2</td>
<td>-61.8±0.4</td>
<td>-81.4±0.1</td>
<td>-80.7±2.3</td>
<td>-61.6±0.4</td>
<td>-63.9±1.0</td>
<td>-66.9±0.6</td>
<td></td>
</tr>
<tr>
<td>Acrobot (SI)*</td>
<td>-387.8±1.0</td>
<td>-169.1±32.3</td>
<td>-156.6±38.9</td>
<td>-233.2±2.6</td>
<td>-216.1±7.7</td>
<td>-170.9±40.3</td>
<td>-250.2±13.7</td>
<td>-245.0±5.5</td>
<td></td>
</tr>
<tr>
<td>Swimmer + Gathering</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>Ant + Gathering</td>
<td>-5.8±5.0</td>
<td>-0.1±0.1</td>
<td>-0.4±0.1</td>
<td>-5.5±0.5</td>
<td>-6.7±0.7</td>
<td>-0.4±0.0</td>
<td>-4.7±0.7</td>
<td>N/A±N/A</td>
<td>-0.3±0.3</td>
</tr>
<tr>
<td>Swimmer + Maze</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>Ant + Maze</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
</tr>
</tbody>
</table>


---
Deep Reinforcement Learning and Control

Multigoal RL

Katerina Fragkiadaki
So far we train one policy/value function per task, e.g., win the game of Tetris, win the game of Go, reach to a *particular* location, put the green cube inside the gray bucket, etc.
Universal value function Approximators

\[ V(s; \theta) \rightarrow V(s, g; \theta) \]

\[ \pi(s; \theta) \rightarrow \pi(s, g; \theta) \]

All the methods we have learnt so far can be used.

At the beginning of an episode, we sample not only a start state but also a goal \( g \), which stays constant throughout the episode.

The experience tuples should contain the goal.

\[ (s, a, r, s') \rightarrow (s, g, a, r, s') \]
Universal value function Approximators

What should be my goal representation?
(not an easy question)

- **Manual**: 3d centroids of objects, robot joint angles and velocities, 3d location of the gripper, etc.
- **Learnt**: We supply a target image as the goal, and the method learns to map it to an embedding vector, e.g., asymmetric actor-critic, Lerrel et al.
Hindsight Experience Replay

Main idea: use failed executions under one goal $g$, as successful executions under an alternative goal $g'$ (which is where we ended spat the end of the episode)

Goal $g$

Our reacher at the end of the episode

$(s, g, a, 0, s')$

Goal $g'$

reward :-)
Hindsight Experience Replay

Marcin Andrychowicz*, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel†, Wojciech Zaremba†
OpenAI

Main idea: use failed executions under one goal $g$, as successful executions under an alternative goal $g'$ (which is where we ended spat the end of the episode)
Hindsight Experience Replay

Algorithm 1: Hindsight Experience Replay (HER)

Given:
- an off-policy RL algorithm $\mathbb{A}$,
- a strategy $\mathbb{S}$ for sampling goals for replay,
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$.

Initialize $\mathbb{A}$
Initialize replay buffer $R$
for episode = 1, $M$ do
  Sample a goal $g$ and an initial state $s_0$.
  for $t = 0, T - 1$ do
    Sample an action $a_t$ using the behavioral policy from $\mathbb{A}$:
    $a_t \leftarrow \pi_b(s_t \| g)$
    Execute the action $a_t$ and observe a new state $s_{t+1}$
  end for
  for $t = 0, T - 1$ do
    $r_t \leftarrow r(s_t, a_t, g)$
    Store the transition $(s_t \| g', a_t, r_t, s_{t+1} \| g)$ in $R$
  end for
  for $g' \in G$ do
    $r' \leftarrow r(s_t, a_t, g')$
    Store the transition $(s_t \| g', a_t, r', s_{t+1} \| g')$ in $R$
  end for
end for
for $t = 1, N$ do
  Sample a minibatch $B$ from the replay buffer $R$
  Perform one step of optimization using $\mathbb{A}$ and minibatch $B$
end for

Usually as additional goal we pick the goal that this episode achieved, and the reward becomes non zero.
Reward shaping: instead of using binary rewards, use continuous rewards, e.g., by considering Euclidean distances from goal configuration.

HER does not require reward shaping! :-)

The burden goes from designing the reward to designing the goal encoding.. :-(
Hindsight Experience Replay

![Graphs showing success rates for different tasks and variations of DDPG](image)

- **DDPG**
- **DDPG+count-based exploration**
- **DDPG+HER**
- **DDPG+HER (version from Sec. 4.5)**

**Tasks**:
- **pushing**
- **sliding**
- **pick-and-place**

**Success Rate**

**Epoch Number** (every epoch = 800 episodes = 800x50 timesteps)