# Differential Privacy in Reinforcement Learning

Waël Doulazmi and Ambroise Odonnat

Master MVA, ENS Paris-Saclay

*name.surname@ens-paris-saclay.fr*

Reinforcement Learning

February 6, 2023

# Overview

# Overview

# Reinforcement Learning

## Definition

Reinforcement Learning is about **agents** learning in an interactive environment using **feedbacks** from their previous actions.

## Principle [Sutton and Barto, 2018]

- State space $\mathcal{S}$, action space $\mathcal{A}$, policy $\pi : \mathcal{S} \mapsto \mathcal{A}$
- $P(s'|s, a)$ transition probability, reward $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- Markov Decision Process (MDP) $M = (\mathcal{S}, \mathcal{A}, R, P)$
- $V^\pi$ value function, $Q^\pi$ action-value function
- Observe $s_t \in \mathcal{S}$, select $a_t \in \mathcal{A}$, receive reward $r_t = R(s_t, a_t)$

The goal is to **maximize** the **expected cumulative rewards**

**Definition**

Reinforcement Learning is about **agents** learning in an interactive environment using **feedbacks** from their previous actions.

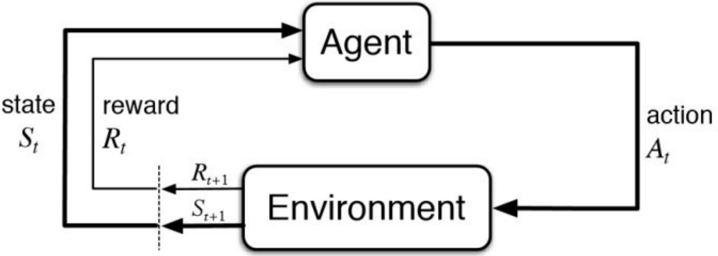



Figure 1: RL loop (source)

# $(\varepsilon, \delta)-$Differential Privacy

## Principle [Dwork et al., 2006]

Mechanism to guarantee that it is **statistically hard** to **infer information** about the environment by observing learned policies.

## Notations

- Privacy budget $\varepsilon$, probability of error $\delta$
- The **higher** the value of $\varepsilon$, the **lower** the privacy
- $\delta$ takes into account bad events

# Central Differential Privacy (CDP)

## Definition [Dwork et al., 2006]

Mechanism $\mathcal{M} : D \mapsto R$ satisfies $(\varepsilon, \delta)$-CDP if for any two adjacent inputs $d, d' \in D$ and for any subset of outputs $S \subseteq R$, it holds that:

$$\mathbb{P}[\mathcal{M}(d) \in S] \leq \exp(\varepsilon)\mathbb{P}[\mathcal{M}(d') \in S] + \delta \tag{1}$$

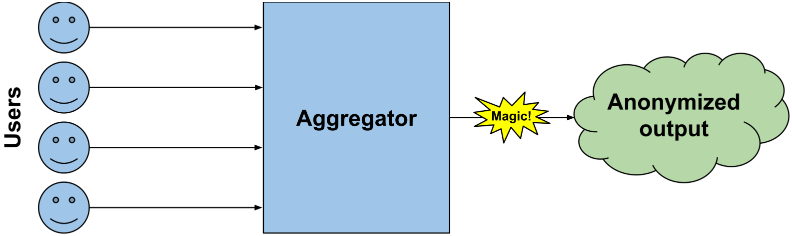



Figure 2: Central Differential Privacy (source)

# Local Differential Privacy (LDP)

## Definition [Duchi et al., 2013]

Mechanism $\mathcal{M}$ satisfies $(\varepsilon, \delta)$-LDP if and only if for all users $u, u' \in \mathcal{U}$, trajectories $(X_u, X_{u'}) \in \mathcal{X}_u \times \mathcal{X}_{u'}$ and all $\mathcal{O} \in \{\mathcal{M}(\mathcal{X}_u) | u \in \mathcal{U}\}$:

$$\mathbb{P}[\mathcal{M}(X_u) \in \mathcal{O}] \leq \exp(\varepsilon)\mathbb{P}[\mathcal{M}(X_{u'}) \in \mathcal{O}] + \delta \tag{2}$$

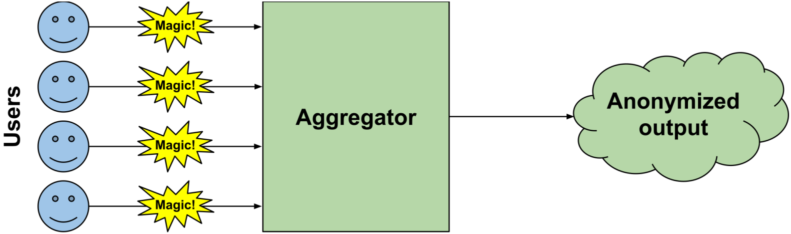



Figure 3: Local Differential Privacy (source)

# Motivation

## Related Work

Several recent papers study private RL algorithms for:

- Tabular RL [Garcelon et al., 2021]
- Bandits [Gajane et al., 2018, Chen et al., 2018]

$\rightarrow$ Lack of private deep RL algorithms with continuous state spaces

## Contributions

- CDP versions of REINFORCE and DQN using DP-SGD
- LDP versions of REINFORCE and DQN
- Study the impact of privacy on the learning process

# Overview

# Cartpole

- Control problem: $\mathcal{S}$ continuous, $\mathcal{A}$ discrete
- Classic benchmark for RL algorithms
- 4-dimensional state $\longrightarrow$ low computational cost

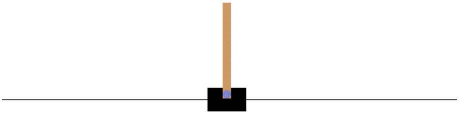


Figure 4: Cartpole (OpenAI Gym)

# Cartpole

## MDP

- $\mathcal{A} = \{0, 1\}$ (push to the left / right)
- $\mathcal{S} \subset \mathbb{R}^4$ (position / speed of the cart and the pole)
- $r(s, a) = 1$ if the pole didn't fail, 0 else

# Cartpole

## MDP

- $\mathcal{A} = \{0, 1\}$ (push to the left / right)
- $\mathcal{S} \subset \mathbb{R}^4$ (position / speed of the cart and the pole)
- $r(s, a) = 1$ if the pole didn't fail, 0 else

## Episode

- $\tau = (s_0, a_o, r_0, s_1, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$
- $s_0 \sim \mu_0$ (uniform)
- Termination when:
  - $\star$ Pole fails
  - $\star$ Reach 500 steps

# Cartpole

## MDP

- $\mathcal{A} = \{0, 1\}$ (push to the left / right)
- $\mathcal{S} \subset \mathbb{R}^4$ (position / speed of the cart and the pole)
- $r(s, a) = 1$ if the pole didn't fail, 0 else

## Episode

- $\tau = (s_0, a_o, r_0, s_1, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$
- $s_0 \sim \mu_0$ (uniform)
- Termination when:
  - ⋆ Pole fails
  - ⋆ Reach 500 steps

$\rightarrow$ Solved when average reward for 25 consecutive trials $\geq 475$

# REINFORCE

## Policy Approximation

$$\pi \in \mathcal{F}_\pi = \left\{ \pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1],\ \forall s \in \mathcal{S},\ \sum_{a \in \mathcal{A}} \pi_\theta(s, a) = 1,\ \theta \in \mathbb{R}^d \right\}$$

# REINFORCE

## Policy Approximation

$$\pi \in \mathcal{F}_\pi = \left\{ \pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow [0,1],\ \forall s \in \mathcal{S},\ \sum_{a \in \mathcal{A}} \pi_\theta(s,a) = 1,\ \theta \in \mathbb{R}^d \right\}$$

## Criterion - Policy performance

$$\max_{\theta \in \mathbb{R}^d} J(\pi_\theta), \qquad J(\pi_\theta) = \mathbb{E}_\tau \left[ \sum_{t=0}^{H-1} \gamma^t r_t \,|\, a_t \sim \pi_\theta(s_t), s_0 \sim \mu_0 \right]$$

# REINFORCE

## Policy Approximation

$$\pi \in \mathcal{F}_\pi = \left\{ \pi_\theta : \mathcal{S} \times \mathcal{A} \to [0,1],\ \forall s \in \mathcal{S},\ \sum_{a \in \mathcal{A}} \pi_\theta(s,a) = 1,\ \theta \in \mathbb{R}^d \right\}$$

## Criterion - Policy performance

$$\max_{\theta \in \mathbb{R}^d} J(\pi_\theta), \qquad J(\pi_\theta) = \mathbb{E}_\tau \left[ \sum_{t=0}^{H-1} \gamma^t r_t \,\middle|\, a_t \sim \pi_\theta(s_t), s_0 \sim \mu_0 \right]$$

## Policy Gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\tau \left[ \left( \sum_{t=0}^{H-1} \nabla_\theta \log(\pi_\theta(s_t, a_t)) \right) \left( \sum_{t=0}^{H-1} \gamma^t r_t \right) \right]$$

## Implementation

- 2 layers perceptron, $h = 128$, $\mathrm{dropout} = 0.5$
- Estimate gradient with one episode

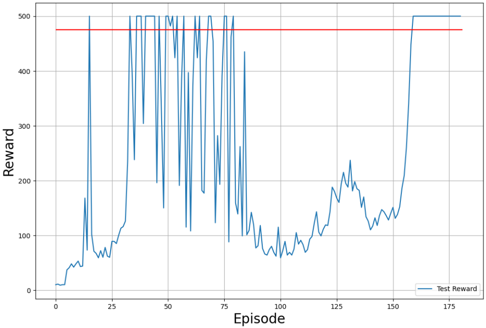



Figure 5: Convergence of REINFORCE on Cartpole

# DQN

## Value Approximation

$$Q \in \mathcal{F}_Q = \{Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R},\ \theta \in \mathbb{R}^d\}$$

# DQN

## Value Approximation

$$Q \in \mathcal{F}_Q = \{Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R},\ \theta \in \mathbb{R}^d\}$$

## Criterion - Temporal Difference

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta), \qquad \mathcal{L}(\theta) = ||Q_\theta(s,a) - (r + \gamma \max_{a'} Q_\theta(s', a'))||_2$$

# DQN

## Value Approximation

$$Q \in \mathcal{F}_Q = \{Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R},\ \theta \in \mathbb{R}^d\}$$

## Criterion - Temporal Difference

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta), \qquad \mathcal{L}(\theta) = ||Q_\theta(s, a) - (r + \gamma \max_{a'} Q_\theta(s', a'))||_2$$

## Deep Q-Network

- Generate a transition using $\pi^{\varepsilon}_{Q_\theta}$ ($\varepsilon$-greedy), add to buffer $\mathcal{D}$
- Sample batch $\mathcal{B} \subset \mathcal{D}$, perform SGD

# DQN

## Implementation

- 3 layers perceptron, $h = 128$
- $|\mathcal{B}| = 128$ transitions

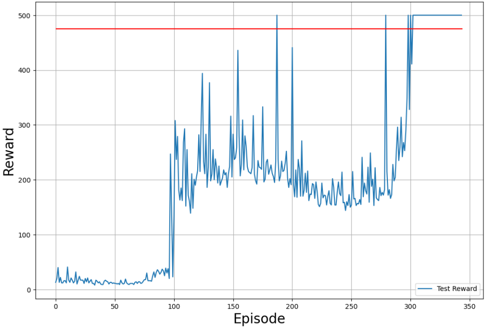



Figure 6: Convergence of DQN on Cartpole

# Overview

# DP-SGD

## Principle [Abadi et al., 2016]

- Differentially private implementation of SGD
- AWGN noise on gradients of each sample during training
- Notable efforts to keep fast and parallel computation

## Challenges

- Adapt noise strategy for REINFORCE and DQN
- Custom backpropagation for DQN

**Input:** Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate $\eta_t$, noise scale $\sigma$, group size $L$, gradient norm bound $C$.

**Initialize** $\theta_0$ randomly

**for** $t \in [T]$ **do**

Take a random sample $L_t$ with sampling probability $L/N$

**Compute gradient**

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

**Clip gradient**

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

**Add noise**

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

**Descent**

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

**Output** $\theta_T$ and compute the overall privacy cost $(\varepsilon, \delta)$ using a privacy accounting method.

Figure 7: DP-SGD

# REINFORCE - Add noise on gradients

```
optimizer.zero_grad()
returns = returns.detach()
loss = - (returns * log_prob_actions).sum()
loss.backward()  # Fills p.grad
# Add noise to the gradients
for p in policy.parameters():
    clip_grad_norm_(p, max_grad_norm)

    # Gaussian noise mechanism
    p.grad += torch.normal(mean=0.,
                           std=sigma * max_grad_norm,
                           size=p.size())

# Optimize model
optimizer.step()
```

# DQN - Custom Backpropagation

```python
# Functional to compute gradient for each sample of a batch
ft_compute_grad = grad(compute_loss_stateless_model)
ft_compute_sample_grad = vmap(ft_compute_grad,
                              in_dims=(None, None, 0, 0,
                                       0, None, None, None))

# Obtain functional version of model, params and buffers
fmodel, params, buffers = make_functional_with_buffers(policy_net)

# Recover gradients of all params for each sample of a batch
sample_grads = ft_compute_sample_grad(params, buffers, state_batch,
                                      action_batch, expected_Q_values,
                                      max_grad_norm, fmodel, criterion)
```

# DQN - Add noise on gradients

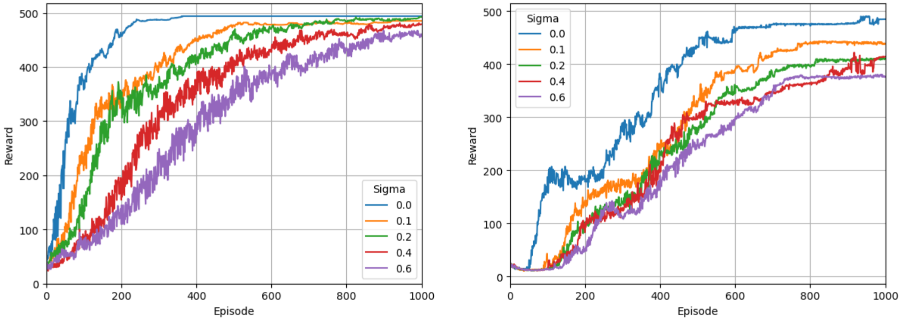
```python
for i, param in enumerate(policy_net.parameters()):

    # Clip inplace gradients seperately
    # for each sample of the batch
    sample_grad = sample_grads[i]
    torch.nn.utils.clip_grad_norm_(sample_grad,
                                   max_norm=max_grad_norm)

    # Aggregate gradients to have a unique grad for the batch
    param.grad = torch.mean(sample_grad, dim=0)

    # Add noise
    noise = torch.normal(mean=0, std=sigma*max_grad_norm,
                         size=param.size()).to(device)
    param.grad += noise/batch_size # Gaussian noise

# Optimize model
optimizer.step()
```

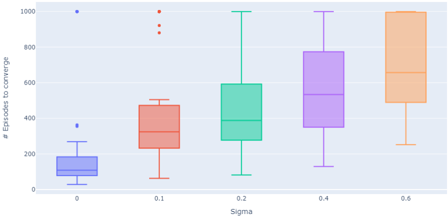



(a) REINFORCE

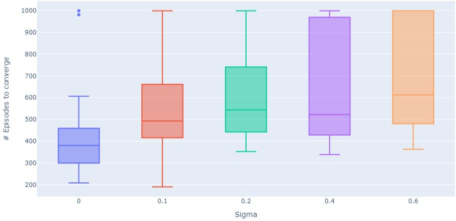
(b) DQN

Figure 8: Impact of noise level on the average agent's reward

# CDP - Results




(a) REINFORCE




(b) DQN

Figure 9: Impact of noise level on the convergence time

# Overview

## Method

- Disturb observations before they are seen by the agent
- Observations are the states and rewards
- Add gaussian noise on the states observed by agents

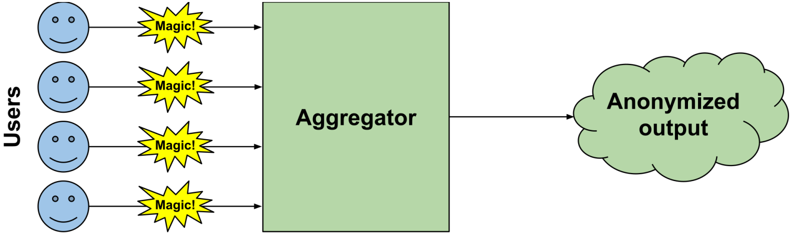



Figure 10: Local Differential Privacy (source)

# LDP - Noise calibration

## Cartpole

- Reward signal is discrete and indicates episode end
- $s \in \mathbb{R}^4$, all its component have a different range

$\longrightarrow$ Noise $\mathcal{N}(0_4, \sigma \times IQR)$

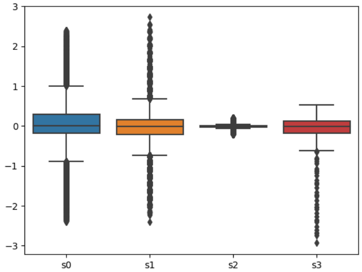
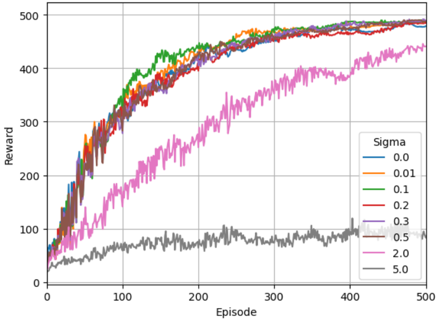
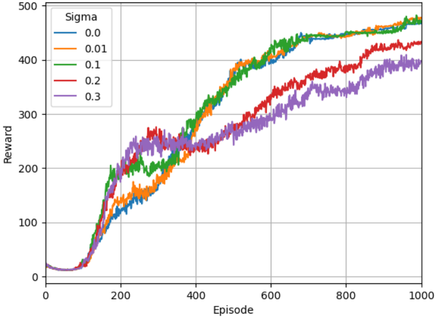



Figure 11: Evolution of $s$ during one training

# LDP - Results



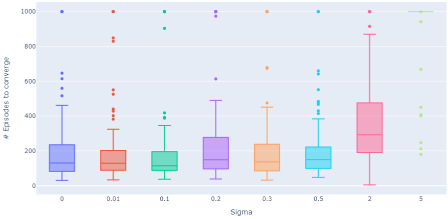

(a) REINFORCE



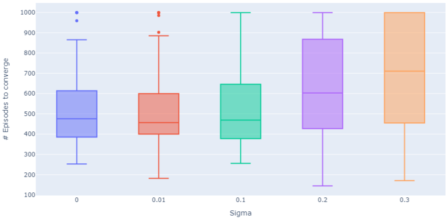

(b) DQN

Figure 12: Impact of noise level on the average agent's reward

# LDP - Results




(a) REINFORCE

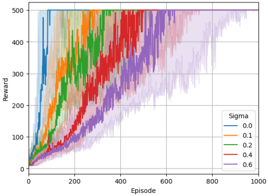



(b) DQN

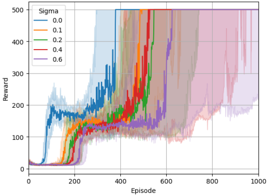
Figure 13: Impact of noise level on the convergence time

# Overview

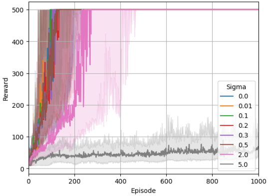
# Summary



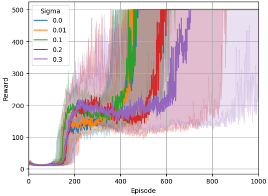

(a) DP-SGD REINFORCE




(b) DP-SGD DQN




(c) REINFORCE with LDP




(d) DQN with LDP

Figure 14: Impact of noise level on the median agent's reward

# Conclusion

## Interesting results

- Implemented DP versions of two deep RL algorithms
- Proof of concept on the Cartpole environment
- Privacy vs Sample-efficiency trade-off
- Algorithms are able to converge with reasonable noise levels

## Further work

- Need for theoretical guarantees on privacy
- Explore scalability on larger environments

Thanks for your attention !

# References


Sutton and Barto (2018)
Reinforcement Learning: An Introduction
*The MIT Press*

Dwork et al. (2006)
Calibrating Noise to Sensitivity in Private Data Analysis
*Springer Berlin Heidelberg*

Duchi et al. (2013)
Local Privacy, Data Processing Inequalities, and Statistical Minimax Rates
*Advances in Neural Information Processing Systems*

Garcelon et al. (2021)
Local Differential Privacy for Regret Minimization in Reinforcement Learning
*Advances in Neural Information Processing Systems*

# References


Gajane et al. (2018)
Corrupt Bandits for Preserving Local Privacy
*Proceedings of Machine Learning Research*

Chen et al. (2018)
(Locally) Differentially Private Combinatorial Semi-Bandits
*International Conference on Machine Learning*

Abdadi et al. (2016)
Deep Learning with Differential Privacy
*Association for Computing Machinery*