Scalable Deep Reinforcement Learning Algorithms for Mean Field Games

Sarah Perrin
sarah.perrin@inria.fr

Joint work with: Mathieu Laurière, Sertan Girgin, Paul Muller, Ayush Jain, Theophile Cabannes, Georgios Piliouras, Julien Pérolat, Romuald Élie, Olivier Pietquin, Matthieu Geist

39th International Conference on Machine Learning

NYU 上海 SHANGHAI 纽约大学 · Université de Lille · Google Research · DeepMind · Berkeley University of California · Singapore University of Technology and Design

Motivations: MFG + RL?

- **Mean Field Game (MFG) for Multi-agent reinforcement learning (MARL)**
  - Efficiently approximate games with very large population of agents
  - Goal: Scale up MARL in terms of number of agents

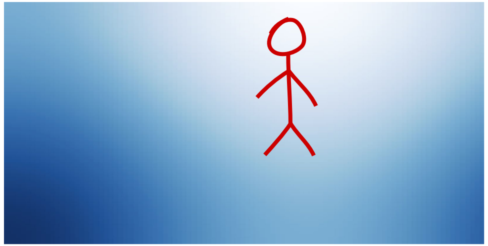
- **Reinforcement learning (RL) for MFG**
  - RL has been successful at solving very complex optimal control problems
  - Goal: Scale up MFGs in terms of model complexity

Mean Field Approximation

Where should I put my towel?

Mean field approximation

Density of people around me



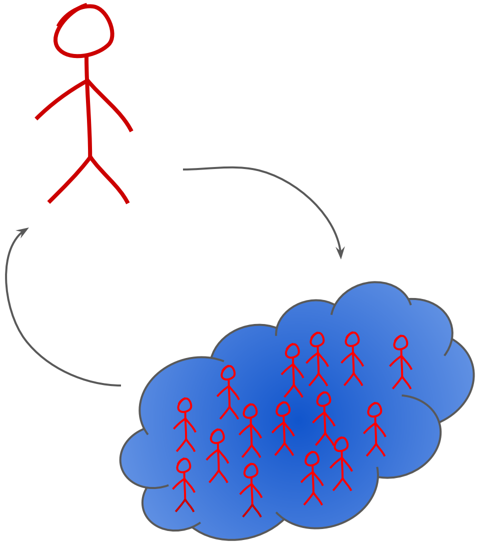
[Image credit: Unsplash]

- **Continuum of infinitesimal players**
  - Homogeneity
  - Symmetry
- **Mean Field Nash equilibrium:**
  - Representative player
  - Population μ *(mean field)*

⇒ Simpler representation

⇒ Approximate equilibrium for finite games

Mean Field Game (MFG)



Mean Field Nash Equilibrium:

- Individual optimization: Best response against the population

$$J(\pi, \mu^*) \leq J(\pi^*, \mu^*) \qquad \text{for all policies } \pi$$

- Population consistency: Everyone uses π^*:

$$\mu^* = \text{distribution induced by } \pi^*$$

Learning in MFGs: basic approaches

- **Basic idea:** iteratively
    - Given the mean field μ, update the policy π
        - **Best response** against the current population distribution
        - Greedy improvement after **evaluating** the previous policy
    - Given policy π, update the mean field μ

Learning in MFGs: basic approaches

- **Basic idea:** iteratively
  - Given the mean field μ, update the policy π
    - **Best response** against the current population distribution
    - Greedy improvement after **evaluating** the previous policy
  - Given policy π, update the mean field μ
- **Reinforcement Learning** can be used for the policy update
  - To compute an optimal policy or to evaluate a policy
  - MDP parameterized by μ

Learning in MFGs: basic approaches

- **Basic idea:** iteratively
  - Given the mean field μ, update the policy π
    - **Best response** against the current population distribution
    - Greedy improvement after **evaluating** the previous policy
  - Given policy π, update the mean field μ
- **Reinforcement Learning** can be used for the policy update
  - To compute an optimal policy or to evaluate a policy
  - MDP parameterized by μ
- **Issue:** lack of convergence in many cases (oscillations, ...)

Learning in MFGs: regularization

- **Improvements:**
    - Averaging past policies or mean fields (e.g., exponential smoothing, ...)
    - Regularizing the policies (e.g., π = softmax(Q) instead of π = argmax(Q))
    - Regularizing the rewards (e.g., entropic regularization, ...)

- Two typical examples:
    - **Fictitious Play (FP)**: fixed point iterations & average past policies / mean fields
    - **Online Mirror Descent (OMD)**: policy evaluation iterations & sum Q-values & policy regu.

Learning in MFGs: regularization

- **Improvements:**
  - Averaging past policies or mean fields (e.g., exponential smoothing, ...)
  - Regularizing the policies (e.g., softmax instead of argmax)
  - Regularizing the rewards (e.g., entropic regularization, ...)
- Two typical examples:
  - **Fictitious Play (FP)**: fixed point iterations & average past policies / mean fields
  - **Online Mirror Descent (OMD)**: policy evaluation iterations & sum Q-values & policy regu.
- **Question:** *How to do this with deep neural networks?*

  (More generally: how to sum two parameterized functions that depends non-linearly on the parameters?)

Learning in MFGs: regularization

- **Improvements:**
  - Averaging past policies or mean fields (e.g., exponential smoothing, ...)
  - Regularizing the policies (e.g., softmax instead of argmax)
  - Regularizing the rewards (e.g., entropic regularization, ...)

- Two typical examples:
  - **Fictitious Play (FP)**: fixed point iterations & average past policies / mean fields
  - **Online Mirror Descent (OMD)**: policy evaluation iterations & sum Q-values & policy regu.

- **Question:** *How to do this with deep neural networks?*

  (More generally: how to sum two parameterized functions that depends non-linearly on the parameters?)

- **Our contribution:** Two scalable algorithms: D-AFP & D-MOMD

Algorithm 1: Deep Average-network Fictitious Play

Algorithm 1: D-AFP

Initialize an empty reservoir buffer $\mathcal{M}_{SL}$ for supervised learning of average policy
Initialize parameters $\bar{\theta}^0$
for $k = 1, \ldots, K$ **do**
  1. Distribution: generate $\bar{\mu}^k$ with $\bar{\pi}_{\bar{\theta}^{k-1}}$
  2. BR: Train $\hat{\pi}_{\theta^k}$ against $\bar{\mu}^{k-1}$, *e.g.* using DQN
  Collect $N_{samples}$ state-action using $\hat{\pi}_{\theta^k}$ and add them to $\mathcal{M}_{SL}$
  3. Average policy: Update $\bar{\pi}_{\bar{\theta}^k}$ by adjusting $\bar{\theta}^k$ (through gradient descent) to minimize:

$$\mathcal{L}(\bar{\theta}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}} \left[-\log \left(\bar{\pi}_{\bar{\theta}}(a|s) \right) \right],$$

  where $\bar{\pi}_{\bar{\theta}}$ is the neural net policy with parameters $\bar{\theta}$
end for
Return $\bar{\mu}^K, \bar{\pi}_{\bar{\theta}^K}$

Builds upon Neural-Fictitious Self Play [Heinrich & Silver, 2016]

Algorithm 2: Deep Munchausen Online Mirror Descent

Algorithm 2: D-MOMD

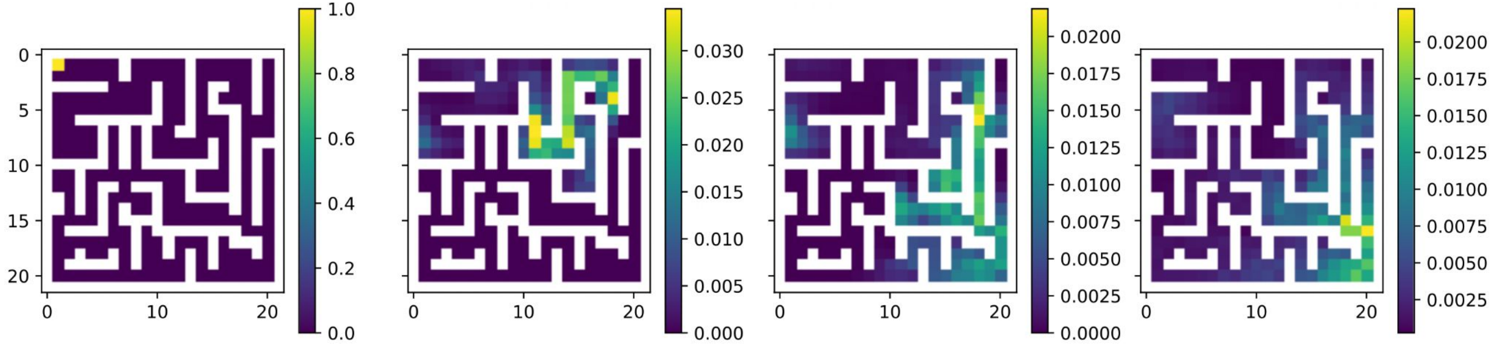
Input: Munchausen parameters τ and α; numbers of OMD iterations K and DQN estimation iterations L
Output: cumulated Q value function, policy π
Initialize the parameters θ^0
Set $\pi^0(a|(n,x)) = \text{softmax}\left(\frac{1}{\tau}\check{Q}_{\theta^0}((n,x),\cdot)\right)(a)$
for $k = 1, \dots, K$ **do**
  1. Distribution: Generate μ^k with π^{k-1}
  2. Value function: Initialize θ^k
  for $\ell = 1, \dots, L$ **do**
    Sample a minibatch of N_B transitions: $\left\{\left((n_i,x_i), a_i, r_{n_i}(x_i,a_i,\mu^k_{n_i}), (n_i+1, x'_i)\right)\right\}_{i=1}^{N_B}$ with $n_i \leq N_T$, $x'_i \sim p_{n_i}(\cdot|x_i,a_i,\mu^k_{n_i})$ and a_i is chosen by an $\epsilon-$greedy policy based on $\check{Q}_{\theta^k}$
    Update θ^k with one gradient step of:
    $$\theta \mapsto \frac{1}{N_B}\sum_{i=1}^{N_B}\left|\check{Q}_\theta((n_i,x_i),a_i) - T_i\right|^2$$
    where T_i is the target defined above
  end for
  3. Policy: for all n, x, a, let
  $$\pi^k(a|(n,x)) = \text{softmax}\left(\frac{1}{\tau}\check{Q}_{\theta^k}((n,x),\cdot)\right)(a)$$
end for
Return $\check{Q}_{\theta^K}, \pi^K$

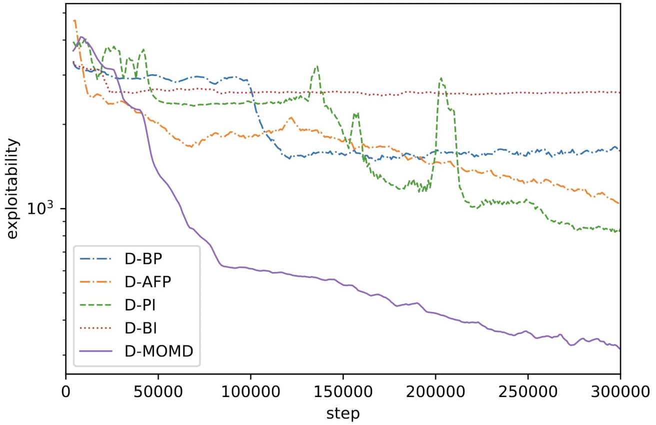
Builds upon Munchausen trick [Vieillard, Pietquin & Geist, 2020]

Numerical Experiments

- **Implementation:** OpenSpiel [Lanctot et al., 2019]
  - Open source
  - Wide range of games (including MFGs) and algorithms
- **5 Models:** typical examples of MFGs are illustrated:
  - SIS
  - Linear-Quadratic
  - Exploration
  - Crowd modeling with congestion
  - Multi-population chasing
- **3 Baselines:** D-AFP and D-MOMD are compared with:
  - Deep Fixed Point (D-FP)
  - Deep Policy Iteration (D-PI)
  - Deep Boltzmann Iteration (D-BI) from [Cui & Koeppl, 2021]

Numerical Example





Main take-away:

- D-AFP and D-MOMD converge even when other baselines fail to converge
- D-MOMD converges faster

Thank you for your attention

Meet me at **Poster Session 1**, **6:30pm**