Numerical Methods for Mean Field Games

Lecture 6
Reinforcement Learning Methods

Mathieu LAURIÈRE

New York University Shanghai

UM6P Vanguard Center, Université Cadi AYYAD,
University Côte d'Azur, & GE2MI
Open Doctoral Lectures
July 5 – 7, 2023

Outline

1. Introduction

2. RL for MFC (MFRL)

3. RL for MFGs

4. MFGs in OpenSpiel

5. Conclusion

Motivations

- In the methods discussed so far, the algorithm uses the full knowledge of the model
  - to write the ODEs or PDEs (lectures 2, 3 and 5)
  - to write the FBSDEs (lecture 4)
  - to compute the gradient in the direct approach (lecture 4)

- Can we learn the solution without using the full knowledge the model and by instead relying on a simulator? $\rightarrow$ model-free reinforcement learning (RL)

- Motivations
  - sometimes we really do not know the model and we only have a simulator (e.g., nature)
  - sometimes we do know the model, but using an exact method is too costly (e.g., very large spaces / complex models)

Motivations

(Reinforcement) Learning in games: many recent successes, e.g.:

Go [Silver et al., 2016, Silver et al., 2017, Silver et al., 2018], Chess [Campbell et al., 2002], Checkers [Schaeffer et al., 2007], Hex [Anthony et al., 2017], Starcraft II [Vinyals et al., 2019], poker games [Brown and Sandholm, 2017, Brown and Sandholm, 2019, Moravčík et al., 2017, Bowling et al., 2015], Stratego [McAleer et al., 2020], [Perolat et al., 2022] . . .

Motivations

(Reinforcement) Learning in games: many recent successes, e.g.:

Go [Silver et al., 2016, Silver et al., 2017, Silver et al., 2018], Chess [Campbell et al., 2002], Checkers [Schaeffer et al., 2007], Hex [Anthony et al., 2017], Starcraft II [Vinyals et al., 2019], poker games [Brown and Sandholm, 2017, Brown and Sandholm, 2019, Moravčík et al., 2017, Bowling et al., 2015], Stratego [McAleer et al., 2020], [Perolat et al., 2022] . . .

Motivations for combining RL and MFGs:

- Scaling up **population size** $\rightarrow$ **Mean Field Games**
- Scaling up **environment complexity** $\rightarrow$ (model-free) **Reinforcement Learning**

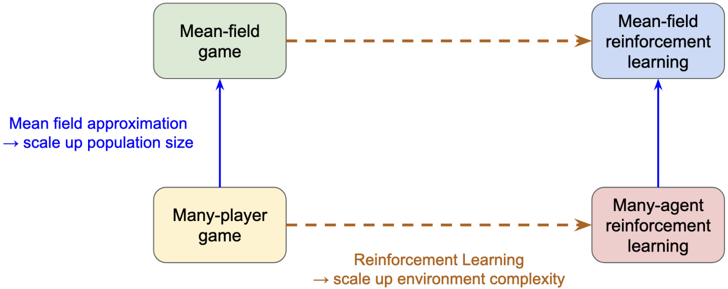
Motivations

(Reinforcement) Learning in games: many recent successes, e.g.:

Go [Silver et al., 2016, Silver et al., 2017, Silver et al., 2018], Chess [Campbell et al., 2002], Checkers [Schaeffer et al., 2007], Hex [Anthony et al., 2017], Starcraft II [Vinyals et al., 2019], poker games [Brown and Sandholm, 2017, Brown and Sandholm, 2019, Moravčík et al., 2017, Bowling et al., 2015], Stratego [McAleer et al., 2020], [Perolat et al., 2022] . . .

Motivations for combining RL and MFGs:

- Scaling up **population size** → **Mean Field Games**
- Scaling up **environment complexity** → (model-free) **Reinforcement Learning**



Reinforcement Learning – Setup

- **Markov Decision Process (MDP):** $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:
  - $\mathcal{S}$: state space, $\mathcal{A}$: action space,
  - $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$: transition kernel, $p(\cdot|s,a)$ gives next state's distribution
  - $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: reward function, $\gamma \in (0,1)$: discount factor
- **Goal:** Find (stationary, mixed) policy π^* : $\mathcal{S} \to \mathcal{P}(\mathcal{A})$ maximizing:

$$R(\pi) = \mathbb{E}\left[\sum_{n\geq 0} \gamma^n r(s_n, a_n)\right], \qquad \text{with } a_n \sim \pi(\cdot|s_n), s_{n+1} \sim p(\cdot|s_n, a_n)$$

Reinforcement Learning – Setup

- **Markov Decision Process (MDP):** $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:
  - $\mathcal{S}$: state space, $\mathcal{A}$: action space,
  - $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$: transition kernel, $p(\cdot|s, a)$ gives next state's distribution
  - $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: reward function, $\gamma \in (0, 1)$: discount factor
- **Goal:** Find (stationary, mixed) policy $\pi^* : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maximizing:

$$R(\pi) = \mathbb{E}\left[\sum_{n\geq 0} \gamma^n r(s_n, a_n)\right], \qquad \text{with } a_n \sim \pi(\cdot|s_n), s_{n+1} \sim p(\cdot|s_n, a_n)$$

- **Model:** p, r

Reinforcement Learning – Setup

- **Markov Decision Process (MDP):** $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:
  - $\mathcal{S}$: state space, $\mathcal{A}$: action space,
  - $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$: transition kernel, $p(\cdot|s, a)$ gives next state's distribution
  - $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: reward function, $\gamma \in (0, 1)$: discount factor
- **Goal:** Find (stationary, mixed) policy $\pi^* : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maximizing:

$$R(\pi) = \mathbb{E}\left[\sum_{n \geq 0} \gamma^n r(s_n, a_n)\right], \qquad \text{with } a_n \sim \pi(\cdot|s_n), s_{n+1} \sim p(\cdot|s_n, a_n)$$

- **Model:** p, r
- **Two settings:**

  (1) **Known model** : Optimal control theory & methods

  (2) **Sample transitions & rewards**: Reinforcement Learning (RL) framework

We want to **learn** the best control by performing **experiments** of the form:

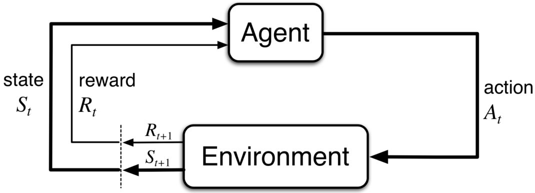
Given the current state S_t,
(1) *Take an action* A_t
(2) *Observe reward* R_{t+1} *& new state* S_{t+1}

We want to **learn** the best control by performing **experiments** of the form:

Given the current state S_t*,*

(1) *Take an action* A_t

(2) *Observe reward* R_{t+1} *& new state* S_{t+1}



Source: [Sutton and Barto, 2018]

- **Learning the policy:**
  - Policy Gradient

$$\theta^{(\mathtt{k}+1)} = \theta^{(\mathtt{k})} - \eta^{(\mathtt{k})} \nabla J(\theta^{(\mathtt{k})}), \qquad \pi^{(\mathtt{k})}(a|s) = \pi(s|a, \theta^{(\mathtt{k})})$$

- **Learning the policy:**
  - Policy Gradient

$$\theta^{(\mathtt{k}+1)} = \theta^{(\mathtt{k})} - \eta^{(\mathtt{k})} \nabla J(\theta^{(\mathtt{k})}), \qquad \pi^{(\mathtt{k})}(a|s) = \pi(s|a, \theta^{(\mathtt{k})})$$

  - PPO, TRPO
  - ...

- **Learning the policy:**
  - Policy Gradient

$$\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta^{(\mathtt{k})} \nabla J(\theta^{(\mathtt{k})}), \qquad \pi^{(\mathtt{k})}(a|s) = \pi(s|a, \theta^{(\mathtt{k})})$$

  - PPO, TRPO
  - ...
- **Learning the value function:**
  - Q-learning

$$Q^*(s, a) = r(s, a) + \gamma \max_{\pi} \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')} \left[Q^*(s', a') \right]$$

Note: $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$, $\alpha^*(s) = \mathrm{argmax}_{a \in \mathcal{A}} Q^*(s, a)$

- **Learning the policy:**
  - Policy Gradient

$$\theta^{(k+1)} = \theta^{(k)} - \eta^{(k)} \nabla J(\theta^{(k)}), \qquad \pi^{(k)}(a|s) = \pi(s|a, \theta^{(k)})$$

  - PPO, TRPO
  - ...
- **Learning the value function:**
  - Q-learning

$$Q^*(s,a) = r(s,a) + \gamma \max_{\pi} \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')} \left[Q^*(s', a') \right]$$

    Note: $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$, $\alpha^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s,a)$
  - Deep Q-neural network (DQN)
  - ...

Reinforcement Learning – Methods

- **Learning the policy:**
  - Policy Gradient

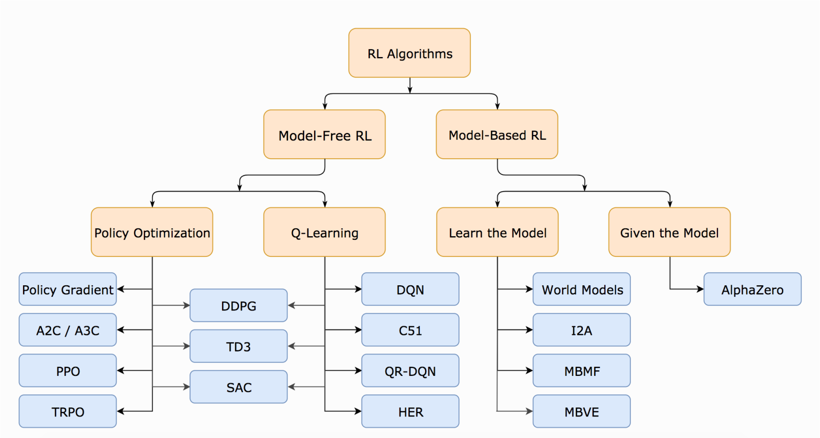
$$\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta^{(\mathtt{k})} \nabla J(\theta^{(\mathtt{k})}), \qquad \pi^{(\mathtt{k})}(a|s) = \pi(s|a, \theta^{(\mathtt{k})})$$

  - PPO, TRPO
  - ...
- **Learning the value function:**
  - Q-learning

$$Q^*(s, a) = r(s, a) + \gamma \max_{\pi} \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')} \left[Q^*(s', a') \right]$$

  Note: $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$, $\alpha^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a)$
  - Deep Q-neural network (DQN)
  - ...
- **Hybrid:**
  - Deep Deterministic Policy Gradient (DDPG)
  - Soft Actor Critic (SAC)
  - ...

RL Taxonomy



Source: [OpenAI Spinning Up][1]

[1] https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

DQN

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Source: [Mnih et al., 2013]

DDPG

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' 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 $\mathcal{N}$ 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:

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

    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

Source: [Lillicrap et al., 2016]

SAC

Algorithm 1 Soft Actor-Critic

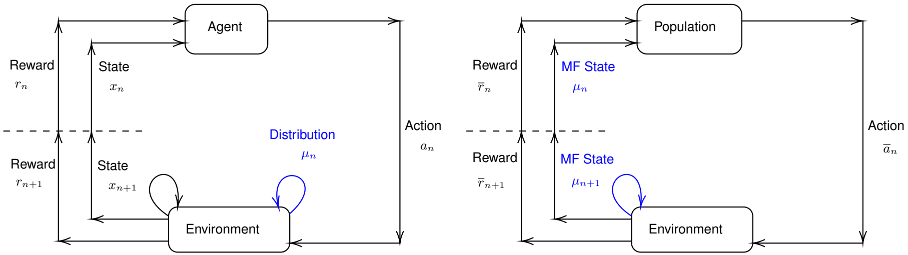
Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

Source: [Haarnoja et al., 2018]

RL Setting for MFG and MFC

Intuitively:

- MFG: a representative agent learns by interacting with an environment, which depends on the population distribution
- MFC: the whole population learns



Population Distribution Approximation

How to deal with the population distribution μ?

- Empirical distribution μ^N
- Histogram (discrete state space)
- ϵ-net in $\mathcal{P}(\mathcal{X})$
- Function approximation for the density:
  - Kernels
  - Neural nets: normalizing flows, ...
  - ...
- ...

So far, most of the literature on RL for MFGs focuses on finite state space models

But see e.g. [Perrin et al., 2021b] in continuous space using normalizing flows

A (Non-exhaustive) Glance at the literature: RL for MFG

- MARL with mean field approximation: [Yang et al., 2018]
- Inverse RL: [Yang et al., 2017], [Chen et al., 2021]
- Multi-time scales: [Subramanian and Mahajan, 2019a], [Angiuli et al., 2020b, Angiuli et al., 2020a, Angiuli and Hu, 2021]
- Fictitious Play with tabular RL: [Perrin et al., 2020], with deep RL: [Elie et al., 2020a, Cui and Koeppl, 2021a] and distribution embedding: [Perrin et al., 2021c]
- Fixed point iterations with Q-learning and variants: [Guo et al., 2019a, Guo et al., 2020], [Anahtarci et al., 2019, Anahtarcı et al., 2021], [Xie et al., 2021]
- Entropy regularization: [Anahtarci et al., 2020a], [Cui and Koeppl, 2021a]
- LQ MFG with actor-Critic: [Fu et al., 2019, uz Zaman et al., 2020], or policy gradient: [Wang et al., 2021]
- RL for partially observable MFG: [Subramanian et al., 2020b]
- Mean field RL for multiple types: [Subramanian et al., 2020a, uz Zaman et al., 2022]
- Learning Master policies with deep RL: [Perrin et al., 2021a]
- ...

A (Non-exhaustive) Glance at the literature: RL for MFC

- Early works on MDP viewpoint: [Gast and Gaujal, 2011, Gast et al., 2012a]
- Policy optimization for stationary MFC: [Subramanian and Mahajan, 2019a]
- Policy gradient for LQ MFC [Carmona et al., 2019a, Wang et al., 2021] and zero sum mean field type game [Carmona et al., 2020]
- Multi-time scale for MFC (and MFG): [Angiuli et al., 2020b, Angiuli et al., 2020a, Angiuli and Hu, 2021]:
- Mean field MDP: dynamic programming and RL [Carmona et al., 2019b, Gu et al., 2019, Motte and Pham, 2019a, Gu et al., 2020a, Cui et al., 2021]
- Decentralized network approach [Gu et al., 2021]
- Model based RL for MFC: [Pasztor et al., 2021]
- ...

Several talks on this topic are available here:
`https://sites.google.com/view/mlmfgseminar/past-talks`

Survey on this topic: [Laurière et al., 2022a] (updated version soon)

Intuitively, at least 3 different settings:

- Static:
  - **No states** (normal-form game): each player chooses an **action** $a \sim \pi(\cdot)$
  - Reward: depends on own action & population's action distribution
  - Examples: towel on the beach, urban settlement, ...
- Stationary:
  - **Infinite horizon**: learns a **stationary policy** $\pi(\cdot|x)$
  - Reward: similar than Evolutive case.
  - Initial state distribution = stationary distribution induced by the population's policy or gamma discounted distribution.
  - Examples: player joining a crowd already in a steady state
- Evolutive:
  - **(In)Finite horizon**: each player learns a **time-dependant policy** $\pi_n(\cdot|x)$
  - Reward: depends on own state, action & population's (state,action) distribution.
  - Fixed initial state distribution
  - Examples: crowd motion, traffic routing, ...
- Other settings: asymptotic, γ-discounted, ergodic, ...

In the sequel we mostly stick to the evolutive setting.

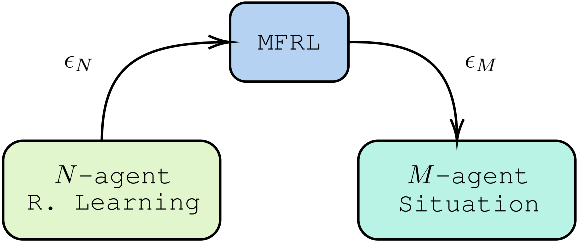
Outline

Outline

From Optimal Control to MFRL

$$
\begin{tikzcd}
\text{Mean-Field Control} \arrow[rr, "\textit{unknown model}"] & & \text{Mean-Field Reinforcement Learning} \\
\text{Optimal Control} \arrow[u, "\substack{N\to\infty \\ MKV}"'] \arrow[rr, "\textit{unknown model}"'] & & \text{Reinforcement Learning} \arrow[u, "\substack{N\to\infty \\ MKV}"]
\end{tikzcd}
$$

Mean Field Control: Finite Population Approximation



Dynamics and cost

- **Dynamics:** discrete time

$$X_{n+1}^{\alpha,\mu} = F(X_n^{\alpha,\mu}, \alpha_n, \mu_n, \epsilon_{n+1}, \epsilon_{n+1}^0), \quad n \geq 0, \qquad X_0^{\alpha,\mu} \sim \mu_0$$

  - $X_n^{\alpha,\mu} \in \mathcal{S} \subseteq \mathbb{R}^d$: state, $\alpha_n \in \mathcal{U} \subseteq \mathbb{R}^k$: action
  - $\epsilon_n \sim \nu$: idiosyncratic noise, $\epsilon_n^0 \sim \nu^0$: common noise (random env.)
  - $p(x'|x, a, \mu)$: corresponding transition probability distribution
  - $\mu_n \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$: a state-action distribution
  - π_n: a policy; randomized actions: $\alpha_n \sim \pi_n(\cdot|s_n)$ or $\alpha_n \sim \pi_n(\cdot|s_n, \mu_n)$

- **Cost:** $\mathbb{J}(\pi; \mu) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^{\alpha,\mu}, \alpha_n, \mu_n\right)\right]$

Problem Formulation

Two scenarios:

- **Cooperative (MFC):** Find π^* s.t.

$$\pi^* \text{ minimizes } \pi \mapsto J^{MFC}(\pi) = \mathbb{J}(\pi; \mu^\pi) \text{ where } \mu^\pi_n = \mathbb{P}^0_{X_n^{\alpha,\mu^\pi}}$$

- **Non-Cooperative (MFG):** Find $(\hat{\pi}, \hat{\mu})$ s.t.

$$\begin{cases} \hat{\pi} \text{ minimizes } \pi \mapsto J^{MFG}(\pi; \hat{\mu}) = \mathbb{J}(\pi; \hat{\mu}) \\ \hat{\mu}_n = \mathbb{P}^0_{X_n^{\hat{\alpha},\hat{\mu}}} \end{cases}$$

In this section we focus on the MFC case

MFG in the next section

MFMDP with Common Noise & Randomization

- **Key Remark:**

$$\alpha^* \in \underset{\alpha}{\operatorname{argmin}} J^{MFC}(\alpha) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^{\alpha}, \alpha_n, \mu_n^{\pi}\right)\right], \qquad \mu_n^{\pi} = \mathbb{P}^0_{X_n^{\alpha}}$$

MFMDP with Common Noise & Randomization

- **Key Remark:**

$$\alpha^* \in \operatorname*{argmin}_{\alpha} J^{MFC}(\alpha) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^{\alpha}, \alpha_n, \mu_n^{\pi}\right)\right], \qquad \mu_n^{\pi} = \mathbb{P}^0_{X_n^{\alpha}}$$

$$= \mathbb{E}_{\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}\times\mathcal{U}} f\left(x, a, \mu_n^{\pi}\right) \nu_n^{\pi}(dx, da)}_{\text{function of } \nu_n^{\pi}}\right]$$

MFMDP with Common Noise & Randomization

- **Key Remark:**

$$\alpha^* \in \underset{\alpha}{\operatorname{argmin}}\, J^{MFC}(\alpha) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^\alpha, \alpha_n, \mu_n^\pi\right)\right], \qquad \mu_n^\pi = \mathbb{P}^0_{X_n^\alpha}$$

$$= \mathbb{E}_{\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}\times\mathcal{U}} f\left(x, a, \mu_n^\pi\right) \nu_n^\pi(dx, da)}_{\text{function of } \nu_n^\pi}\right]$$

- **Lifted problem:** population / social planner's optimization problem:
  - $\rightarrow$ state = population distribution μ_n^π
  - $\rightarrow$ value function = function of the distribution μ

MFMDP with Common Noise & Randomization

- **Key Remark:**

$$\alpha^* \in \underset{\alpha}{\text{argmin}}\, J^{MFC}(\alpha) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^\alpha, \alpha_n, \mu_n^\pi\right)\right], \qquad \mu_n^\pi = \mathbb{P}^0_{X_n^\alpha}$$

$$= \mathbb{E}_{\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}\times\mathcal{U}} f\left(x, a, \mu_n^\pi\right) \nu_n^\pi(dx, da)}_{\text{function of } \nu_n^\pi}\right]$$

- **Lifted problem:** population / social planner's optimization problem:
  - $\rightarrow$ state = population distribution μ_n^π
  - $\rightarrow$ value function = function of the distribution μ
- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$ with constraint: $pr_1(\bar{a}) = \mu$
  - Transition function: $\mu' = \bar{F}(\mu, \bar{a}, \epsilon^0) \sim \bar{p}(\mu, \bar{a})$
  - Reward function: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{S}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

MFMDP with Common Noise & Randomization

- **Key Remark:**

$$\alpha^* \in \underset{\alpha}{\operatorname{argmin}}\, J^{MFC}(\alpha) = \mathbb{E}_{\epsilon,\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n f\left(X_n^\alpha, \alpha_n, \mu_n^\pi\right)\right], \qquad \mu_n^\pi = \mathbb{P}^0_{X_n^\alpha}$$

$$= \mathbb{E}_{\epsilon^0}\left[\sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}\times\mathcal{U}} f\left(x, a, \mu_n^\pi\right) \nu_n^\pi(dx, da)}_{\text{function of } \nu_n^\pi}\right]$$

- **Lifted problem:** population / social planner's optimization problem:
  - $\rightarrow$ state = population distribution μ_n^π
  - $\rightarrow$ value function = function of the distribution μ
- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{S}\times\mathcal{U})$ with constraint: $pr_1(\bar{a}) = \mu$
  - Transition function: $\mu' = \bar{F}(\mu, \bar{a}, \epsilon^0) \sim \bar{p}(\mu, \bar{a})$
  - Reward function: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{S}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$
- **Goal:** max. $\bar{J}^{\bar{\pi}}(\mu) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n \bar{r}\left(\mu_n^{\bar{\pi}}, \bar{a}_n\right)\right]$, $\bar{a}_n \sim \bar{\pi}(\cdot|\mu_n^{\bar{\pi}})$, $\mu_{n+1}^{\bar{\pi}} \sim \bar{p}(\cdot|\mu_n^{\bar{\pi}}, \bar{a}_n)$,
  $\mu_0^{\bar{\pi}} = \mu$
- **Mean field policy:** $\bar{\pi}$ kernel $\bar{\mathcal{S}} \rightarrow \mathcal{P}(\bar{\mathcal{A}})$, *randomized population-strategies* $\bar{a}$

Dynamic Programming Principle (DPP)

Theorem: DPP for MFMDP [Carmona et al., 2019c]

Under suitable conditions,

$$\bar{J}^*(\mu) := \sup_{\bar{\pi}} \bar{J}^{\bar{\pi}}(\mu) = \sup_{\bar{\pi}} \left\{ \int_{\bar{\mathcal{A}}} \left[\bar{r}(\mu, \bar{a}) + \gamma \mathbb{E}\left[\bar{J}^*\left(\bar{F}(\mu, \bar{a}, \epsilon^0) \right) \right] \right] \bar{\pi}(d\bar{a}|\mu) \right\},$$

where the sup is over a subset of $\{\bar{\pi} : \bar{\mathcal{S}} \to \mathcal{P}(\bar{\mathcal{A}})\}$

Likewise for **mean field state-action value function** $\bar{Q}^*$

Dynamic Programming Principle (DPP)

Theorem: DPP for MFMDP [Carmona et al., 2019c]

Under suitable conditions,

$$\bar{J}^*(\mu) := \sup_{\bar{\pi}} \bar{J}^{\bar{\pi}}(\mu) = \sup_{\bar{\pi}} \left\{ \int_{\bar{A}} \left[\bar{r}(\mu, \bar{a}) + \gamma \mathbb{E}\left[\bar{J}^*\left(\bar{F}(\mu, \bar{a}, \epsilon^0) \right) \right] \right] \bar{\pi}(d\bar{a}|\mu) \right\},$$

where the sup is over a subset of $\{\bar{\pi} : \bar{\mathcal{S}} \to \mathcal{P}(\bar{\mathcal{A}})\}$

Likewise for **mean field state-action value function** $\bar{Q}^*$

Proof: based on “double lifting” [Bertsekas and Shreve, 1996]

Dynamic Programming Principle (DPP)

Theorem: DPP for MFMDP [Carmona et al., 2019c]

Under suitable conditions,

$$\bar{J}^*(\mu) := \sup_{\bar{\pi}} \bar{J}^{\bar{\pi}}(\mu) = \sup_{\bar{\pi}} \left\{ \int_{\bar{\mathcal{A}}} \left[\bar{r}(\mu, \bar{a}) + \gamma \mathbb{E}\left[\bar{J}^*\left(\bar{F}(\mu, \bar{a}, \epsilon^0) \right) \right] \right] \bar{\pi}(d\bar{a}|\mu) \right\},$$

where the sup is over a subset of $\{\bar{\pi} : \bar{\mathcal{S}} \to \mathcal{P}(\bar{\mathcal{A}})\}$

Likewise for **mean field state-action value function** $\bar{Q}^*$

Proof: based on "double lifting" [Bertsekas and Shreve, 1996]

DPPs for MFC: [Laurière and Pironneau, 2016], [Pham and Wei, 2017], [Gast et al., 2012b], [Gu et al., 2020b], [Djete et al., 2019], [Motte and Pham, 2019b], ...

Dynamic Programming Principle (DPP)

Theorem: DPP for MFMDP [Carmona et al., 2019c]

Under suitable conditions,

$$\bar{J}^*(\mu) := \sup_{\bar{\pi}} \bar{J}^{\bar{\pi}}(\mu) = \sup_{\bar{\pi}} \left\{ \int_{\bar{\mathcal{A}}} \left[\bar{r}(\mu, \bar{a}) + \gamma \mathbb{E}\left[\bar{J}^*\left(\bar{F}(\mu, \bar{a}, \epsilon^0) \right) \right] \right] \bar{\pi}(d\bar{a}|\mu) \right\},$$

where the sup is over a subset of $\{\bar{\pi} : \bar{\mathcal{S}} \to \mathcal{P}(\bar{\mathcal{A}})\}$

Likewise for **mean field state-action value function** $\bar{Q}^*$

Proof: based on "double lifting" [Bertsekas and Shreve, 1996]

DPPs for MFC: [Laurière and Pironneau, 2016], [Pham and Wei, 2017], [Gast et al., 2012b], [Gu et al., 2020b], [Djete et al., 2019], [Motte and Pham, 2019b], . . .

Here: discrete time, infinite horizon, common noise, feedback controls, . . .

→ well-suited for **RL**

→ Mean-field Q-learning algorithm

Mean Field Learning Settings

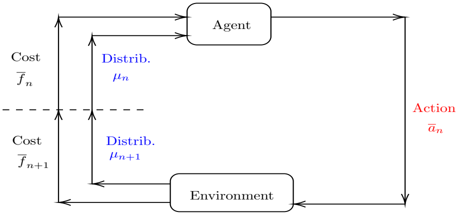
Hierarchy of settings:

- **Setting 1: known model**: computational method based on knowledge of MFMDP
  (a) Gradient based methods
  (b) Dynamic programming based methods

Mean Field Learning Settings

Hierarchy of settings:

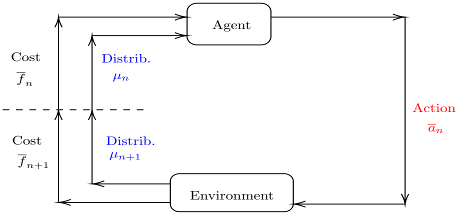
- **Setting 1: known model**: computational method based on knowledge of MFMDP
  - (a) Gradient based methods
  - (b) Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning



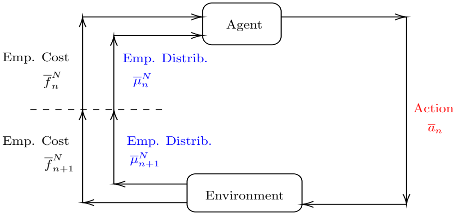
Mean Field Learning Settings

Hierarchy of settings:

- **Setting 1: known model**: computational method based on knowledge of MFMDP
  - (a) Gradient based methods
  - (b) Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning



- **Setting 3:** unknown model but **samples from N-agent MDP**: approx. MF learning



Outline

Algorithm

Idea 1: *Make the "policy gradient" approach model-free*

Policy Gradient (PG) to minimize $J(\theta)$

- Control $\approx$ parameterized function (analog to the "direct approach" in lecture 4)
- Look for the optimal parameter θ^*
- Perform gradient descent on the space of parameters

Algorithm

Idea 1: *Make the "policy gradient" approach model-free*

Policy Gradient (PG) to minimize $J(\theta)$

- Control $\approx$ parameterized function (analog to the "direct approach" in lecture 4)
- Look for the optimal parameter θ^*
- Perform gradient descent on the space of parameters

Hierarchy of three situations, more and more complex:

(1) access to the exact **(mean field) model**: $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla J(\theta^{(k)})$

Algorithm

Idea 1: *Make the "policy gradient" approach model-free*

Policy Gradient (PG) to minimize $J(\theta)$

- Control $\approx$ parameterized function (analog to the "direct approach" in lecture 4)
- Look for the optimal parameter θ^*
- Perform gradient descent on the space of parameters

Hierarchy of three situations, more and more complex:

(1) access to the exact **(mean field) model**: $\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta \nabla J(\theta^{(\mathtt{k})})$

(2) access to a **mean field simulator**:

$\rightarrow$ idem + gradient estimation (0^{th}-order opt.): $\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta \widetilde{\nabla} J(\theta^{(\mathtt{k})})$

Algorithm

Idea 1: *Make the "policy gradient" approach model-free*

Policy Gradient (PG) to minimize $J(\theta)$

- Control $\approx$ parameterized function (analog to the "direct approach" in lecture 4)
- Look for the optimal parameter θ^*
- Perform gradient descent on the space of parameters

Hierarchy of three situations, more and more complex:

(1) access to the exact **(mean field) model**: $\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta \nabla J(\theta^{(\mathtt{k})})$

(2) access to a **mean field simulator**:

$\rightarrow$ idem + gradient estimation (0^{th}-order opt.): $\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta \widetilde{\nabla} J(\theta^{(\mathtt{k})})$

(3) access to a N-agent **population simulator**:

$\rightarrow$ idem + error on mean $\approx$ empirical mean (LLN): $\theta^{(\mathtt{k+1})} = \theta^{(\mathtt{k})} - \eta \widetilde{\nabla}^N J(\theta^{(\mathtt{k})})$

Algorithm

Idea 1: *Make the "policy gradient" approach model-free*

Policy Gradient (PG) to minimize $J(\theta)$

- Control $\approx$ parameterized function (analog to the "direct approach" in lecture 4)
- Look for the optimal parameter θ^*
- Perform gradient descent on the space of parameters

Hierarchy of three situations, more and more complex:

(1) access to the exact **(mean field) model**: $\theta^{(\mathtt{k}+1)} = \theta^{(\mathtt{k})} - \eta \nabla J(\theta^{(\mathtt{k})})$

(2) access to a **mean field simulator**:

$\rightarrow$ idem + gradient estimation (0^{th}-order opt.): $\theta^{(\mathtt{k}+1)} = \theta^{(\mathtt{k})} - \eta \widetilde{\nabla} J(\theta^{(\mathtt{k})})$

(3) access to a N-agent **population simulator**:

$\rightarrow$ idem + error on mean $\approx$ empirical mean (LLN): $\theta^{(\mathtt{k}+1)} = \theta^{(\mathtt{k})} - \eta \widetilde{\nabla}^N J(\theta^{(\mathtt{k})})$

Theorem: For **Linear-Quadratic** MFC [Carmona et al., 2019c]

In each case, convergence holds at a linear rate:

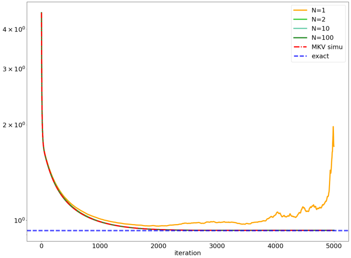
Taking $\mathtt{k} \approx \mathcal{O}\Big(\log(1/\epsilon)\Big)$ is sufficient to ensure $J(\theta^{(\mathtt{k})}) - J(\theta^*) < \epsilon$.

Proof: builds on [Fazel et al., 2018], analysis of perturbation of Riccati equations

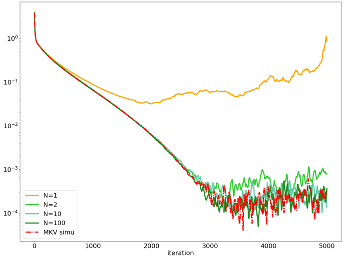
Numerical Illustration

Example: Linear dynamics, quadratic costs of the type:

$$f(x,\mu,\alpha) = \underbrace{(\bar{\mu} - x)^2}_{\text{distance to mean position}} + \underbrace{\alpha^2}_{\text{cost of moving}}, \qquad \bar{\mu} = \underbrace{\int \mu(\xi)\,d\xi}_{\text{mean position}},$$



Value of the MF cost



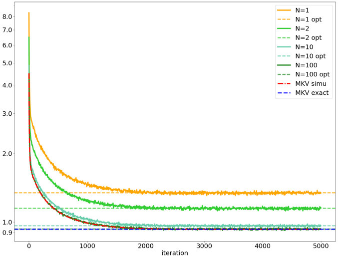
Rel. err. on MF cost

MF cost = cost in the mean field problem

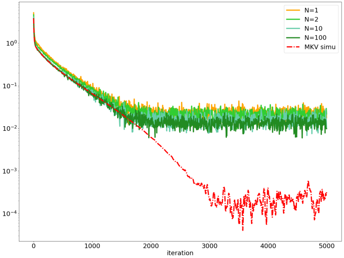
Numerical Illustration

Example: Linear dynamics, quadratic costs of the type:

$$f(x,\mu,\alpha) = \underbrace{(\bar{\mu}-x)^2}_{\text{distance to mean position}} + \underbrace{\alpha^2}_{\text{cost of moving}}, \qquad \bar{\mu} = \underbrace{\int \mu(\xi)d\xi}_{\text{mean position}},$$



Value of the social cost



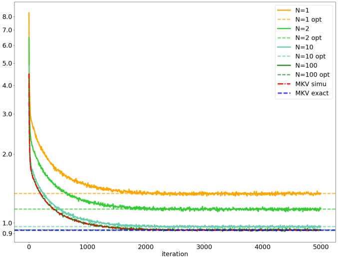
Rel. err. on social cost

Social cost = average over the N-agents

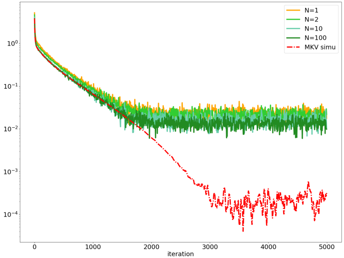
Numerical Illustration

Example: Linear dynamics, quadratic costs of the type:

$$f(x,\mu,\alpha) = \underbrace{(\bar{\mu}-x)^2}_{\text{distance to mean position}} + \underbrace{\alpha^2}_{\text{cost of moving}}, \qquad \bar{\mu} = \underbrace{\int \mu(\xi)d\xi}_{\text{mean position}},$$



Value of the social cost



Rel. err. on social cost

Social cost = average over the N-agents

Main take-away:

Trying to learn the mean-field regime solution can be efficient even for N small

Outline

Mean Field Q-Function

Idea 2: *Generalize Q-learning to Mean-Field Control*

Reminder:

- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$ with constraint: $pr_1(\bar{a}) = \mu$
  - Transition function: $\mu' = \bar{F}(\mu, \bar{a}, \epsilon^0) \sim \bar{p}(\mu, \bar{a})$
  - Reward function: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{S}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

- **Goal:** max. $\bar{J}^{\bar{\pi}}(\mu) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n \bar{r}\left(\mu_n^{\bar{\pi}}, \bar{a}_n\right)\right]$, $\bar{a}_n \sim \bar{\pi}(\cdot|\mu_n^{\bar{\pi}})$, $\mu_{n+1}^{\bar{\pi}} \sim \bar{p}(\cdot|\mu_n^{\bar{\pi}}, \bar{a}_n)$, $\mu_0^{\bar{\pi}} = \mu$

Mean Field Q-Function

Idea 2: *Generalize Q-learning to Mean-Field Control*

Reminder:

- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$ with constraint: $pr_1(\bar{a}) = \mu$
  - Transition function: $\mu' = \bar{F}(\mu, \bar{a}, \epsilon^0) \sim \bar{p}(\mu, \bar{a})$
  - Reward function: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{S}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

- **Goal:** max. $\bar{J}^{\bar{\pi}}(\mu) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n \bar{r}\left(\mu_n^{\bar{\pi}}, \bar{a}_n\right)\right]$, $\bar{a}_n \sim \bar{\pi}(\cdot|\mu_n^{\bar{\pi}})$, $\mu_{n+1}^{\bar{\pi}} \sim \bar{p}(\cdot|\mu_n^{\bar{\pi}}, \bar{a}_n)$,
$\mu_0^{\bar{\pi}} = \mu$

Q-function associated to a policy π:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}\left[Q^\pi(s', a')\right]$$

Mean Field Q-function associated to a mean field policy $\bar{\pi}$:

$$\bar{Q}^{\bar{\pi}}(\bar{s}, \bar{a}) = \bar{r}(\bar{s}, \bar{a}) + \gamma \mathbb{E}_{\bar{s}' \sim \bar{p}(\cdot|\bar{s},\bar{a}), \bar{a}' \sim \bar{\pi}(\cdot|\bar{s}')}\left[\bar{Q}^{\bar{\pi}}(\bar{s}', \bar{a}')\right]$$

MF Q-Learning

- **Optimal MF Q-function:**

$$\bar{Q}^*(\bar{s}, \bar{a}) = \bar{r}(\bar{s}, \bar{a}) + \gamma \sup_{\bar{\pi}} \mathbb{E}_{\bar{a}' \sim \bar{\pi}(\cdot|\bar{s}), \bar{s}' \sim \bar{p}(\cdot|\bar{s}, \bar{a}')} \left[\bar{Q}^*(\bar{s}', \bar{a}') \right]$$

- **Algorithm:**
  - Idealized version (synchronous):

$$\bar{Q}^{(\mathtt{k}+1)}(\bar{s}, \bar{a}) = \bar{r}(\bar{s}, \bar{a}) + \gamma \sup_{\bar{\pi}} \mathbb{E}_{\bar{s}' \sim \bar{p}(\cdot|\bar{s}, \bar{a}), \bar{a}' \sim \bar{\pi}(\cdot|\bar{s}')} \left[\bar{Q}^{(\mathtt{k})}(\bar{s}', \bar{a}') \right], \qquad (\bar{s}, \bar{a}) \in \bar{\mathcal{S}} \times \bar{\mathcal{A}}$$

$$= [\bar{T}^* \bar{Q}^{(\mathtt{k})}](\bar{s}, \bar{a})$$

  - Following a trajectory (async.): $\bar{s}^{(\mathtt{k}+1)} \sim p(\cdot|\bar{s}^{(\mathtt{k})}, \bar{a}^{(\mathtt{k})})$, $\bar{a}^{(\mathtt{k}+1)} \sim \bar{\pi}^{(\mathtt{k}+1)}(\cdot|\bar{s}^{(\mathtt{k})})$,

$$\begin{cases} \bar{Q}^{(\mathtt{k}+1)}(\bar{s}, \bar{a}) = \bar{Q}^{(\mathtt{k})}(\bar{s}, \bar{a}), \qquad (\bar{s}, \bar{a}) \in \bar{\mathcal{S}} \times \bar{\mathcal{A}} \\ \bar{Q}^{(\mathtt{k}+1)}(\bar{s}^{(\mathtt{k}+1)}, \bar{a}^{(\mathtt{k}+1)}) \leftarrow \bar{r}(\bar{s}^{(\mathtt{k}+1)}, \bar{a}^{(\mathtt{k}+1)}) + \gamma \max_{\bar{a}'} \bar{Q}^{(\mathtt{k})}(\bar{s}^{(\mathtt{k}+1)}, \bar{a}') \end{cases}$$

MF Q-Learning

- **Optimal MF Q-function:**

$$\bar{Q}^*(\bar{s},\bar{a}) = \bar{r}(\bar{s},\bar{a}) + \gamma \sup_{\bar{\pi}} \mathbb{E}_{\bar{a}'\sim\bar{\pi}(\cdot|\bar{s}),\bar{s}'\sim\bar{p}(\cdot|\bar{s},\bar{a}')}\left[\bar{Q}^*(\bar{s}',\bar{a}')\right]$$

- **Algorithm:**
  - Idealized version (synchronous):

$$\bar{Q}^{(\mathtt{k}+1)}(\bar{s},\bar{a}) = \bar{r}(\bar{s},\bar{a}) + \gamma \sup_{\bar{\pi}} \mathbb{E}_{\bar{s}'\sim\bar{p}(\cdot|\bar{s},\bar{a}),\bar{a}'\sim\bar{\pi}(\cdot|\bar{s}')}\left[\bar{Q}^{(\mathtt{k})}(\bar{s}',\bar{a}')\right], \qquad (\bar{s},\bar{a}) \in \bar{\mathcal{S}}\times\bar{\mathcal{A}}$$
$$= [\bar{T}^*\bar{Q}^{(\mathtt{k})}](\bar{s},\bar{a})$$

  - Following a trajectory (async.): $\bar{s}^{(\mathtt{k}+1)} \sim p(\cdot|\bar{s}^{(\mathtt{k})},\bar{a}^{(\mathtt{k})})$, $\bar{a}^{(\mathtt{k}+1)} \sim \bar{\pi}^{(\mathtt{k}+1)}(\cdot|\bar{s}^{(\mathtt{k})})$,

$$\begin{cases} \bar{Q}^{(\mathtt{k}+1)}(\bar{s},\bar{a}) = \bar{Q}^{(\mathtt{k})}(\bar{s},\bar{a}), \qquad (\bar{s},\bar{a}) \in \bar{\mathcal{S}}\times\bar{\mathcal{A}} \\ \bar{Q}^{(\mathtt{k}+1)}(\bar{s}^{(\mathtt{k}+1)},\bar{a}^{(\mathtt{k}+1)}) \leftarrow \bar{r}(\bar{s}^{(\mathtt{k}+1)},\bar{a}^{(\mathtt{k}+1)}) + \gamma \max_{\bar{a}'} \bar{Q}^{(\mathtt{k})}(\bar{s}^{(\mathtt{k}+1)},\bar{a}') \end{cases}$$

- **Implementation:** several possibilities (can be combined):
  - pure (population and individual) strategies
  - discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{S}\times\mathcal{U})$
  - deep Reinforcement Learning

MF Q-Learning: Numerical Illustration

Cyber-security example of [Kolokoltsov and Bensoussan, 2016] (see also lecture 5)

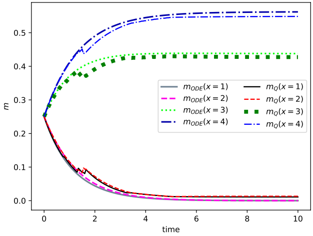
- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$

MF Q-Learning: Numerical Illustration

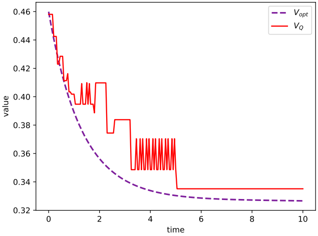
Cyber-security example of [Kolokoltsov and Bensoussan, 2016] (see also lecture 5)

- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$

Test 1: $m_0 = (1/4, 1/4, 1/4, 1/4)$



Evolution of m^{m_0} optimally controlled (m_{ODE}) or controlled using the approximate Q-function (m_Q)



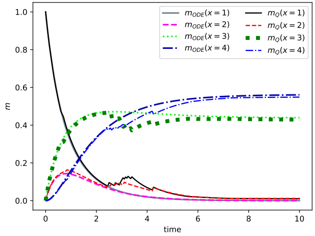
V function (V_{opt}) and approximate Q-function (V_Q) along the optimal flow.

(See section 8.1 of [Laurière, 2021] and section 6.1 of [Carmona et al., 2019c])

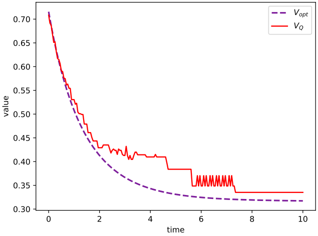
Cyber-security example of [Kolokoltsov and Bensoussan, 2016] (see also lecture 5)

- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$

Test 2: $m_0 = (1, 0, 0, 0)$



Evolution of m^{m_0} optimally controlled (m_{ODE}) or controlled using the approximate Q-function (m_Q)



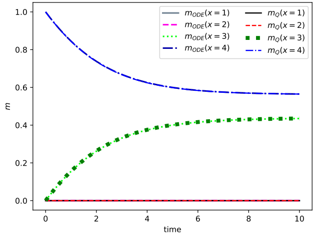
V function (V_{opt}) and approximate Q-function (V_Q) along the optimal flow.

(See section 8.1 of [Laurière, 2021] and section 6.1 of [Carmona et al., 2019c])

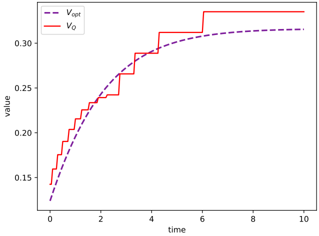
Cyber-security example of [Kolokoltsov and Bensoussan, 2016] (see also lecture 5)

- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{S}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{S} \times \mathcal{U})$

Test 3: $m_0 = (0, 0, 0, 1)$



Evolution of m^{m_0} optimally controlled (m_{ODE}) or controlled using the approximate Q-function (m_Q)



V function (V_{opt}) and approximate Q-function (V_Q) along the optimal flow.

(See section 8.1 of [Laurière, 2021] and section 6.1 of [Carmona et al., 2019c])

Deep RL for MFC

- Instead of discretizing the distribution, we can train a parameterized function to approximate the Q-function
- For instance: neural network trained by DDPG
- Note: We do not need to randomize the policy at the population level, but we do allow randomization at the agent level
- See sections 6.1, 6.2 and 6.3 of [Carmona et al., 2019c]

Sample code

Code

Sample code to illustrate: IPython notebook

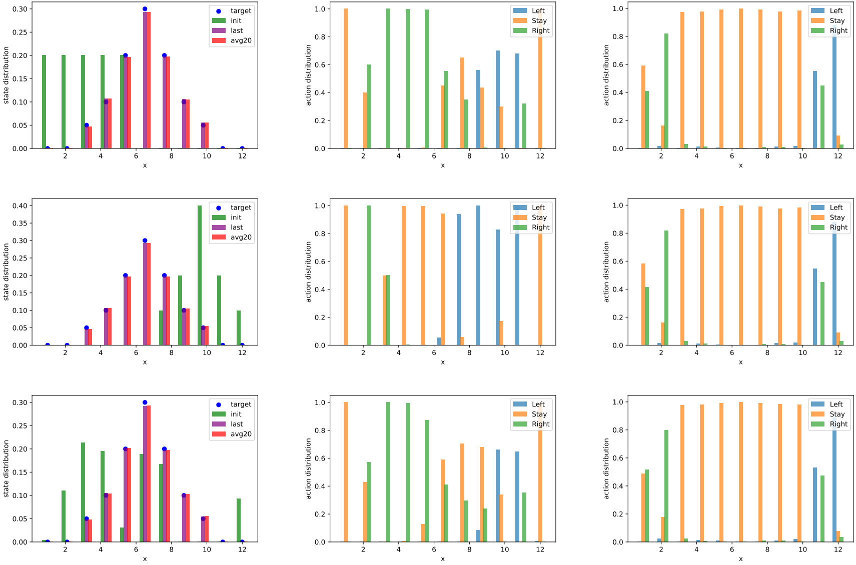
`https://colab.research.google.com/drive/1W8H4EM0bx0RFQFzIaNEcPiEYzG02b0jb?usp=sharing`

- Same example as above: MFC for cybersecurity
- Solved using deep RL with population-dependent controls

Another Example: Distribution Planning

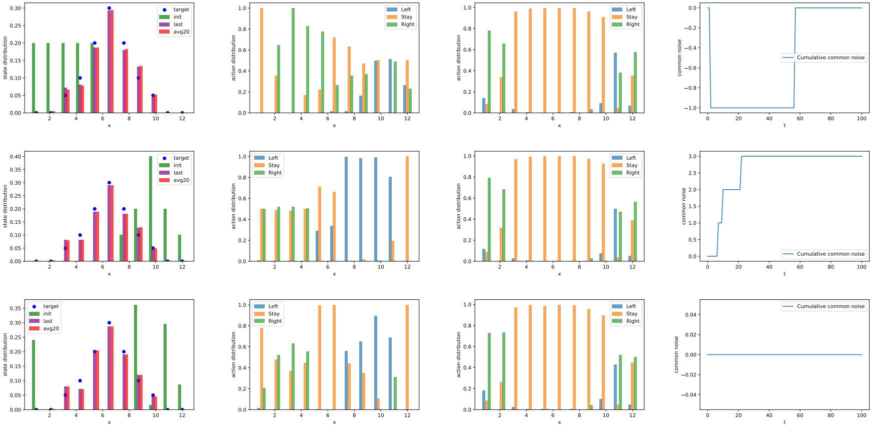
- Goal: match a target distribution.
- $\mathcal{S} = \{1, \dots, 10\}$ and $\mathcal{A} = \{-1, 0, +1\}$.
- Transitions: $F(x, a, \mu, e, e^0) = x + a + e^0$.
- Cost:
$$f(x, a, \mu) = |a| + \sum_i |\mu(i) - \mu_{\text{target}}(i)|^2.$$
- Here we chose: $\mu_{\text{target}} = (0, 0, 0.05, 0.1, 0.2, 0.3, 0.2, 0.1, 0.05, 0, 0)$.
- No idiosyncratic noise.
- Hence in general it is not possible to match the target distribution unless the agents are allowed to randomize their actions at the individual level.
- We use $\mathcal{P}(\mathcal{A})^{\mathcal{S}}$ for the level-1 action space.
- Without or with common noise $\varepsilon_n^0 \in \mathcal{A}$.
- It is not feasible to rely on a tabular method. We show deep RL results.

Another Example: Distribution Planning



More details in [Carmona et al., 2019c]

Another Example: Distribution Planning with Common Noise



More details in [Carmona et al., 2019c]

Outline

The term **“learning”** has many interpretations, such as:

Learning

The term **"learning"** has many interpretations, such as:

- In game theory, economics, . . . :
  [Fudenberg and Levine, 2009]: *"The theory of learning in games [. . .] examines how, which, and what kind of equilibrium might arise as a consequence of a long-run nonequilibrium process of learning, adaptation, and/or imitation"*

Learning

The term **"learning"** has many interpretations, such as:

- In game theory, economics, . . . :
  [Fudenberg and Levine, 2009]: *"The theory of learning in games [. . .] examines how, which, and what kind of equilibrium might arise as a consequence of a long-run nonequilibrium process of learning, adaptation, and/or imitation"*

- In machine learning, RL, . . . :
  [Mitchell et al., 1997]: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."*

Learning/Optimization Algorithms in Games

Learning/optimization methods:

- Fixed point iteration
  - Banach-Picard iterations
  - idem + damping/mixing/smoothing
  - Fictitious Play (FP)
- Online Mirror Descent (OMD)
- ...

Learning/Optimization Algorithms in Games

Learning/optimization methods:

- Fixed point iteration
  - Banach-Picard iterations
  - idem + damping/mixing/smoothing
  - Fictitious Play (FP)
- Online Mirror Descent (OMD)
- ...

in

- Games, particularly in economics, see e.g. [Fudenberg et al., 1998]
- Non-atomic games. see e.g. [Hadikhanloo et al., 2021]
- Mean Field Games, see e.g. [Hadikhanloo, 2018]

Learning in MFGs

Generic structure: repeated game (iterations)

- Update the representative agent behavior
  - value function
  - policy (control)
- Update the population behavior

$$\dots \qquad \mapsto \pi^{(k)} \mapsto \mu^{(k)} \mapsto \pi^{(k+1)} \mapsto \qquad \dots$$

Learning in MFGs

Generic structure: repeated game (iterations)

- Update the representative agent behavior
  - value function
  - policy (control)
- Update the population behavior

$$\dots \qquad \mapsto \pi^{(k)} \mapsto \mu^{(k)} \mapsto \pi^{(k+1)} \mapsto \qquad \dots$$

*Where is there **learning**?*

$\to$ First type of "Learning": meta-algorithm / outside loop

$\to$ Second type of "Learning": agent's viewpoint / inner loop

Best Response and Population Behavior Maps

We focus on MFG and write $J = J^{MFG}$. For simplicity let's forget the common noise.

Two important functions:

- **Best Response map:**

$$\mathtt{BR} : \mu \mapsto \pi \in \operatorname{argmax} J^{MFG}(\cdot; \mu)$$

- **Population Behavior** induced when everyone using a policy:

$$\mathtt{PB} : \pi \mapsto \mu : \mu_{n+1} = \Phi(\mu_n, \pi_n)$$

where:

$$\Phi(\mu, \pi)(x) := \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} p(x|x_0, a, \mu)\pi(a|x_0, \mu)\mu(x_0), \qquad x \in \mathcal{X}$$

represents a one-step transition of the population distribution

Best Response and Population Behavior Maps

We focus on MFG and write $J = J^{MFG}$. For simplicity let's forget the common noise.

Two important functions:

- **Best Response map:**

$$\mathrm{BR} : \mu \mapsto \pi \in \operatorname{argmax} J^{MFG}(\cdot; \mu)$$

- **Population Behavior** induced when everyone using a policy:

$$\mathrm{PB} : \pi \mapsto \mu : \mu_{n+1} = \Phi(\mu_n, \pi_n)$$

where:

$$\Phi(\mu, \pi)(x) := \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} p(x|x_0, a, \mu)\pi(a|x_0, \mu)\mu(x_0), \qquad x \in \mathcal{X}$$

represents a one-step transition of the population distribution

Mean Field Nash equilibrium: $(\hat{\mu}, \hat{\pi})$ such that

$$\begin{cases} \hat{\mu} = \mathrm{PB}(\pi) \\ \hat{\pi} = \mathrm{BR}(\hat{\mu}) \end{cases}$$

$\hat{\mu}$ can be unique without $\hat{\pi}$ being unique!

Outline

Learning in MFGs

Generic structure: repeated game (iterations)

- Update the representative agent behavior
  - value function
  - policy (control)
- Update the population behavior

*Where is there **learning**?*

$\rightarrow$ First type of "Learning": meta-algorithm / outside loop

$\rightarrow$ Second type of "Learning": agent's viewpoint / inner loop

Banach-Picard Fixed Point Iterations

Fixed point method

- Update agent's policy: $\pi^{(\mathtt{k+1})} \in \mathtt{BR}(\mu^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

- **Convergence:** holds under strict contraction property for the map:

$$\mu^{(\mathtt{k})} \mapsto \mu^{(\mathtt{k+1})}$$

Banach-Picard Fixed Point Iterations

Fixed point method

- Update agent's policy: $\pi^{(\mathtt{k+1})} \in \mathtt{BR}(\mu^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

- **Convergence:** holds under strict contraction property for the map:

$$\mu^{(\mathtt{k})} \mapsto \mu^{(\mathtt{k+1})}$$

- Typically ensured by assuming that
  - $\mu^{(\mathtt{k})} \mapsto \pi^{(\mathtt{k+1})}$
  - $\pi^{(\mathtt{k+1})} \mapsto \mu^{(\mathtt{k+1})}$

  are Lipschitz with **small enough** Lipschitz constants

- See e.g. [Huang et al., 2006], [Guo et al., 2019b]

- Can be relaxed with entropy regularization [Anahtarci et al., 2020b], [Cui and Koeppl, 2021b], [Yardim et al., 2022], ...

Banach-Picard Fixed Point Iterations

Fixed point method

- Update agent's policy: $\pi^{(k+1)} \in \mathtt{BR}(\mu^{(k)})$
- Update population's behavior: $\mu^{(k+1)} = \mathtt{Pop}(\pi^{(k+1)})$

- **Convergence:** holds under strict contraction property for the map:

$$\mu^{(k)} \mapsto \mu^{(k+1)}$$

- Typically ensured by assuming that
  - $\mu^{(k)} \mapsto \pi^{(k+1)}$
  - $\pi^{(k+1)} \mapsto \mu^{(k+1)}$

  are Lipschitz with **small enough** Lipschitz constants

- See e.g. [Huang et al., 2006], [Guo et al., 2019b]

- Can be relaxed with entropy regularization [Anahtarci et al., 2020b], [Cui and Koeppl, 2021b], [Yardim et al., 2022], ...

- Can be modified with damping/mixing/smoothing; e.g. Fictitious Play

Fictitious Play method

- Update agent's policy: $\pi^{(\mathtt{k}+1)} \in \mathtt{BR}(\overline{\mu}^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$
- Update population's average behavior: $\overline{\mu}^{(\mathtt{k}+1)} = \frac{\mathtt{k}}{\mathtt{k}+1}\overline{\mu}^{(\mathtt{k}+1)} + \frac{1}{\mathtt{k}+1}\mu^{(\mathtt{k}+1)}$

- **Convergence:** holds under (Lasry-Lions) **monotonicity** structure for the MFG

Fictitious Play method

- Update agent's policy: $\pi^{(\mathtt{k}+1)} \in \mathtt{BR}(\overline{\mu}^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$
- Update population's average behavior: $\overline{\mu}^{(\mathtt{k}+1)} = \frac{\mathtt{k}}{\mathtt{k}+1}\overline{\mu}^{(\mathtt{k}+1)} + \frac{1}{\mathtt{k}+1}\mu^{(\mathtt{k}+1)}$

- **Convergence:** holds under (Lasry-Lions) **monotonicity** structure for the MFG
- Typically ensured by assuming that:
  - p is independent of μ
  - r is separable: $r(x, a, \mu) = r(x, a) + \tilde{r}(x, \mu)$
  - $\tilde{r}$ is monotone: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

Fictitious Play method

- Update agent's policy: $\pi^{(\mathtt{k+1})} \in \mathtt{BR}(\overline{\mu}^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$
- Update population's average behavior: $\overline{\mu}^{(\mathtt{k+1})} = \frac{\mathtt{k}}{\mathtt{k+1}} \overline{\mu}^{(\mathtt{k+1})} + \frac{1}{\mathtt{k+1}} \mu^{(\mathtt{k+1})}$

- **Convergence:** holds under (Lasry-Lions) **monotonicity** structure for the MFG
- Typically ensured by assuming that:
  - p is independent of μ
  - r is separable: $r(x, a, \mu) = r(x, a) + \tilde{r}(x, \mu)$
  - $\tilde{r}$ is monotone: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

- Consequence: the exploitability is a Lyapunov function
- where the **exploitability** of π facing μ is:

$$\mathcal{E}(\pi; \mu) = \sup J(\cdot; \mu) - J(\pi; \mu) \geq 0$$

Fictitious Play method

- Update agent's policy: $\pi^{(k+1)} \in \mathrm{BR}(\overline{\mu}^{(k)})$
- Update population's behavior: $\mu^{(k+1)} = \mathrm{Pop}(\pi^{(k+1)})$
- Update population's average behavior: $\overline{\mu}^{(k+1)} = \frac{k}{k+1}\overline{\mu}^{(k+1)} + \frac{1}{k+1}\mu^{(k+1)}$

- **Convergence:** holds under (Lasry-Lions) **monotonicity** structure for the MFG
- Typically ensured by assuming that:
  - p is independent of μ
  - r is separable: $r(x, a, \mu) = r(x, a) + \tilde{r}(x, \mu)$
  - $\tilde{r}$ is monotone: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

- Consequence: the exploitability is a Lyapunov function
- where the **exploitability** of π facing μ is:

$$\mathcal{E}(\pi; \mu) = \sup J(\cdot; \mu) - J(\pi; \mu) \geq 0$$

- See e.g., [Cardaliaguet and Hadikhanloo, 2017], [Hadikhanloo and Silva, 2019], [Elie et al., 2020b], [Perrin et al., 2020], [Geist et al., 2022], ...

Value iteration VS policy iteration

Reminder:

Fixed point method

- Update agent's policy: $\pi^{(k+1)} \in \mathtt{BR}(\mu^{(k)})$
- Update population's behavior: $\mu^{(k+1)} = \mathtt{Pop}(\pi^{(k+1)})$

- Requires computation of a best response $\Rightarrow$ fully solving an MDP
- This is analogous to value iteration
- An alternative method is policy iteration: greedy update & evaluation

Value iteration VS policy iteration

Reminder:

Fixed point method

- Update agent's policy: $\pi^{(\mathtt{k}+1)} \in \mathtt{BR}(\mu^{(\mathtt{k})})$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$

- Requires computation of a best response $\Rightarrow$ fully solving an MDP
- This is analogous to value iteration
- An alternative method is policy iteration: greedy update & evaluation
- Note: these are not "standard" VI and PI because we need to intertwine updates of the mean field and the policy/value function

Policy iteration

Policy iteration method

- Update agent's Q-function: $Q^{(\mathtt{k+1})} = Q_{\pi^{(\mathtt{k})},\mu^{(\mathtt{k})}}$
- Update agent's policy: $\pi^{(\mathtt{k+1})}(x) = \mathrm{argmax}_{a \in \mathcal{A}} Q^{(\mathtt{k+1})}(x, a), x \in \mathcal{X}$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

- where the representative agent's Q-function, given μ, is:

$$Q_{\pi,\mu}(x, a)$$

$$= \mathbb{E}\left[\sum_{n \geq 0} \gamma^n r(x_n, a_n, \mu)\right], \; x_{n+1} \sim p(\cdot|x_n, a_n, \mu), a_{n+1} \sim \pi(\cdot|x_{n+1}), x_0 = x, a_0 = a$$

$$= r(x, a, \mu) + \gamma \mathbb{E}[Q_{\pi,\mu}(x', a')], \quad x' \sim p(\cdot|x, a, \mu), a' \sim \pi(\cdot|x')$$

Policy iteration method

- Update agent's Q-function: $Q^{(\mathtt{k+1})} = Q_{\pi^{(\mathtt{k})}, \mu^{(\mathtt{k})}}$
- Update agent's policy: $\pi^{(\mathtt{k+1})}(x) = \operatorname{argmax}_{\pi \in \Pi} \langle Q^{(\mathtt{k+1})}(x, \cdot), \pi \rangle, x \in \mathcal{X}$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

- where the representative agent's Q-function, given μ, is:

$$
\begin{aligned}
&Q_{\pi,\mu}(x,a) \\
&= \mathbb{E}\left[\sum_{n \geq 0} \gamma^n r(x_n, a_n, \mu)\right], \; x_{n+1} \sim p(\cdot | x_n, a_n, \mu), a_{n+1} \sim \pi(\cdot | x_{n+1}), x_0 = x, a_0 = a \\
&= r(x, a, \mu) + \gamma \mathbb{E}[Q_{\pi,\mu}(x', a')], \quad x' \sim p(\cdot | x, a, \mu), a' \sim \pi(\cdot | x')
\end{aligned}
$$

- Note: Here, no need to compute a BR; just evaluate a Q function & argmax
- See [Cacace et al., 2021], , [Camilli and Tang, 2022], [Tang and Song, 2022], [Laurière et al., 2023] in the continuous setting, and [Cui and Koeppl, 2021b] in the discrete setting.

Policy iteration

Policy iteration method

- Update agent's Q-function: $Q^{(\mathtt{k+1})} = Q_{\pi^{(\mathtt{k})}, \mu^{(\mathtt{k})}}$
- Update agent's policy: $\pi^{(\mathtt{k+1})}(x) = \operatorname{argmax}_{\pi \in \Pi} \langle Q^{(\mathtt{k+1})}(x, \cdot), \pi \rangle, x \in \mathcal{X}$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

- where the representative agent's Q-function, given μ, is:

$$
\begin{aligned}
&Q_{\pi,\mu}(x, a) \\
&= \mathbb{E}\left[\sum_{n \geq 0} \gamma^n r(x_n, a_n, \mu)\right], \; x_{n+1} \sim p(\cdot | x_n, a_n, \mu), a_{n+1} \sim \pi(\cdot | x_{n+1}), x_0 = x, a_0 = a \\
&= r(x, a, \mu) + \gamma \mathbb{E}[Q_{\pi,\mu}(x', a')], \quad x' \sim p(\cdot | x, a, \mu), a' \sim \pi(\cdot | x')
\end{aligned}
$$

- Note: Here, no need to compute a BR; just evaluate a Q function & argmax
- See [Cacace et al., 2021], , [Camilli and Tang, 2022], [Tang and Song, 2022], [Laurière et al., 2023] in the continuous setting, and [Cui and Koeppl, 2021b] in the discrete setting.
- The updates can be "smoothed" by averaging → Online Mirror Descent

Online Mirror Descent method

- Update agent's Q-function: $Q^{(\mathtt{k+1})} = Q_{\pi^{(\mathtt{k})},\mu^{(\mathtt{k})}}$
- Update agent's average Q-function: $\overline{Q}^{(\mathtt{k+1})} = \overline{Q}^{(\mathtt{k})} + \eta Q^{(\mathtt{k+1})}$
- Update agent's policy by mirroring: $\pi^{(\mathtt{k+1})}(\cdot|x) = \Gamma\left(\overline{Q}^{(\mathtt{k+1})}(x,\cdot)\right)$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

Online Mirror Descent method

- Update agent's Q-function: $Q^{(\mathtt{k}+1)} = Q_{\pi^{(\mathtt{k})},\mu^{(\mathtt{k})}}$
- Update agent's average Q-function: $\overline{Q}^{(\mathtt{k}+1)} = \overline{Q}^{(\mathtt{k})} + \eta Q^{(\mathtt{k}+1)}$
- Update agent's policy by mirroring: $\pi^{(\mathtt{k}+1)}(\cdot|x) = \Gamma\left(\overline{Q}^{(\mathtt{k}+1)}(x,\cdot)\right)$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$

where

$$\Gamma(y) := \nabla h^*(y) = \underset{p\in\mathcal{P}(\mathcal{A})}{\operatorname{argmax}}[\langle y, p\rangle - h(\pi)].$$

with a regularizer $h : \mathcal{P}(\mathcal{A}) \to \mathbb{R}$ and $h^* : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ its convex conjugate defined by $h^*(y) = \max_{p\in\mathcal{P}(\mathcal{A})} [\langle y, p\rangle - h(\pi)]$

Online Mirror Descent method

- Update agent's Q-function: $Q^{(\mathtt{k}+1)} = Q_{\pi^{(\mathtt{k})},\mu^{(\mathtt{k})}}$
- Update agent's average Q-function: $\overline{Q}^{(\mathtt{k}+1)} = \overline{Q}^{(\mathtt{k})} + \eta Q^{(\mathtt{k}+1)}$
- Update agent's policy by mirroring: $\pi^{(\mathtt{k}+1)}(\cdot|x) = \Gamma\left(\overline{Q}^{(\mathtt{k}+1)}(x,\cdot)\right)$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$

where

$$\Gamma(y) := \nabla h^*(y) = \underset{p \in \mathcal{P}(\mathcal{A})}{\operatorname{argmax}}[\langle y, p\rangle - h(\pi)].$$

with a regularizer $h : \mathcal{P}(\mathcal{A}) \to \mathbb{R}$ and $h^* : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ its convex conjugate defined by $h^*(y) = \max_{p \in \mathcal{P}(\mathcal{A})} [\langle y, p\rangle - h(\pi)]$

- **Convergence:** typically under **monotonicity** structure

Online Mirror Descent

Online Mirror Descent method

- Update agent's Q-function: $Q^{(\mathtt{k}+1)} = Q_{\pi^{(\mathtt{k})}, \mu^{(\mathtt{k})}}$
- Update agent's average Q-function: $\overline{Q}^{(\mathtt{k}+1)} = \overline{Q}^{(\mathtt{k})} + \eta Q^{(\mathtt{k}+1)}$
- Update agent's policy by mirroring: $\pi^{(\mathtt{k}+1)}(\cdot|x) = \Gamma\left(\overline{Q}^{(\mathtt{k}+1)}(x, \cdot)\right)$
- Update population's behavior: $\mu^{(\mathtt{k}+1)} = \mathtt{Pop}(\pi^{(\mathtt{k}+1)})$

where

$$\Gamma(y) := \nabla h^*(y) = \operatorname*{argmax}_{p \in \mathcal{P}(\mathcal{A})} [\langle y, p \rangle - h(\pi)].$$

with a regularizer $h : \mathcal{P}(\mathcal{A}) \to \mathbb{R}$ and $h^* : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ its convex conjugate defined by $h^*(y) = \max_{p \in \mathcal{P}(\mathcal{A})} [\langle y, p \rangle - h(\pi)]$

- **Convergence:** typically under **monotonicity** structure
- Note: Here, no need to compute a BR; just evaluate a Q function & argmax

Online Mirror Descent method

- Update agent's Q-function: $Q^{(\mathtt{k+1})} = Q_{\pi^{(\mathtt{k})}, \mu^{(\mathtt{k})}}$
- Update agent's average Q-function: $\overline{Q}^{(\mathtt{k+1})} = \overline{Q}^{(\mathtt{k})} + \eta Q^{(\mathtt{k+1})}$
- Update agent's policy by mirroring: $\pi^{(\mathtt{k+1})}(\cdot|x) = \Gamma\left(\overline{Q}^{(\mathtt{k+1})}(x, \cdot)\right)$
- Update population's behavior: $\mu^{(\mathtt{k+1})} = \mathtt{Pop}(\pi^{(\mathtt{k+1})})$

where

$$\Gamma(y) := \nabla h^*(y) = \operatorname*{argmax}_{p \in \mathcal{P}(\mathcal{A})} [\langle y, p \rangle - h(\pi)].$$

with a regularizer $h : \mathcal{P}(\mathcal{A}) \to \mathbb{R}$ and $h^* : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ its convex conjugate defined by $h^*(y) = \max_{p \in \mathcal{P}(\mathcal{A})} [\langle y, p \rangle - h(\pi)]$

- **Convergence:** typically under **monotonicity** structure
- Note: Here, no need to compute a BR; just evaluate a Q function & argmax
- See e.g., [Hadikhanloo, 2018] in the continuous setting, and [Pérolat et al., 2022], [Geist et al., 2022], . . . in the discrete setting

Summary for FP and OMD

Algorithm: Fixed point iter.

input : Initial policy π^0

$\mu^0 := \mu^{\pi^0}$;
for $k = 1, \ldots, K$: **do**
  $\pi^k :=$ BR against μ^{k-1};
  $\mu^k := \mu^{\pi^k}$;
return π^K, μ^K

$\downarrow$

Algorithm: Fictitious Play

input : Initial policy π^0

$\bar{\pi}^0 := \pi^0$;
$\bar{\mu}^0 := \mu^{\bar{\pi}^0}$;
for $k = 1, \ldots, K$: **do**
  $\pi^k :=$ BR against $\bar{\mu}^{k-1}$;
  $\bar{\mu}^k := \frac{k}{k+1}\bar{\mu}^{k-1} + \frac{1}{k+1}\mu^{\pi^k}$;
  $\bar{\pi}^k :=$ policy giving $\bar{\mu}^k$;
return $\bar{\pi}^K$, $\bar{\mu}^K$

Algorithm: Policy iter.

input : Initial policy π^0

$\mu^0 := \mu^{\pi^0}$;
for $k = 1, \ldots, K$: **do**
  $Q^k :=$ Q-func. for π^{k-1} given μ^{k-1};
  $\pi^k := \operatorname{argmax} Q^k$;
  $\mu^k := \mu^{\pi^k}$;
return π^K, μ^K

$\downarrow$

Algorithm: OMD

input : Initial policy π^0

$\mu^0 := \mu^{\pi^0}$;
for $k = 1, \ldots, K$: **do**
  $Q^k :=$ Q-func. for π^{k-1} given μ^{k-1};
  $\bar{Q}^k := \bar{Q}^{k-1} + \alpha Q^k$;
  $\pi^k := \mathrm{softmax}_\tau\, \bar{Q}^k$;
  $\mu^k := \mu^{\pi^k}$;
return π^K, μ^K

Other Variations and improvements

Possible ways to fix lack of convergence issues:

- Damping / smoothing: e.g.,

$$\mu^{k+1} \leftarrow \text{average of past mean fields}, \pi^{k+1} \leftarrow \text{average of past BR}, \ldots$$

- Softmax policy, e.g.

$$\operatorname{argmax} Q(x, \cdot) \leftarrow \operatorname{softmax}_\tau Q(x, \cdot)$$

- Entropy regularization, e.g.

$$r(x, a, \mu) \leftarrow r(x, a, \mu) - \eta \log\left(\frac{\pi(a|x)}{\tilde{\pi}(a|x)}\right)$$

- ...

$\rightarrow$ Encompasses many possible variants

Outline

1. Introduction

2. RL for MFC (MFRL)

3. RL for MFGs
- Setting
- Learning/Optimization Methods
- Reinforcement Learning Methods
- Unifying RL for MFC and MFG: a Two Timescale Approach

4. MFGs in OpenSpiel

5. Conclusion

Learning in MFGs

Generic structure: repeated game (iterations)

- Update the representative agent behavior
  - value function
  - policy (control)
- Update the population behavior

*Where is there **learning**?*

$\rightarrow$ First type of “Learning”: meta-algorithm / outside loop

$\rightarrow$ Second type of “Learning”: agent’s viewpoint / inner loop

Learning in MFGs

Generic structure: repeated game (iterations)

- Update the representative agent behavior
  - value function
  - policy (control)

- Update the population behavior

*Where is there **learning**?*

$\rightarrow$ First type of "Learning": meta-algorithm / outside loop

$\rightarrow$ Second type of "Learning": agent's viewpoint / inner loop

Given the mean field, the problem faced by a representative player is a *standard* MDP

$\Rightarrow$ We can use any RL algorithm from the literature

Next, we provide some examples

Systemic Risk

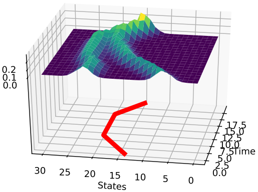
Example (Systemic risk model of [Carmona et al., 2015])

$$J((a_n)_n; (m_n)_n) = -\mathbb{E}\left[\sum_{n=0}^{N_T} \left(a_n^2 \underbrace{-q a_n (m_n - X_n)}_{\substack{\text{borrow if } X_n < m_n \\ \text{lend if } X_n > m_n}} + \kappa (m_n - X_n)^2 \right) + c(m_{N_T} - X_{N_T})^2 \right]$$

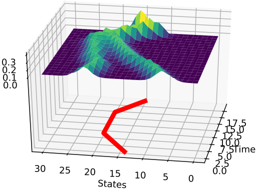
Subj. to: $X_{n+1} = X_n + [K(m_n - X_n) + a_n] + \epsilon_{n+1} + \epsilon^0_{n+1}$

At equilibrium: $m_n = \mathbb{E}[X_n | \epsilon^0], n \geq 0$

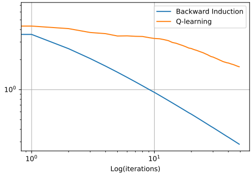
[Perrin et al., 2020]: Fictitious Play with Backward Induction or tabular Q-learning



Exact solution



Fictitious Play & RL



Exploitability

Example (Ergodic crowd aversion model of [Almulla et al., 2017])

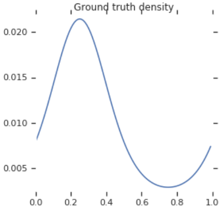
MFG on $\mathbb{T}$,

$$f(x, m, \alpha) = \frac{1}{2}|\alpha|^2 + \tilde{f}(x) + \ln(m(x)),$$

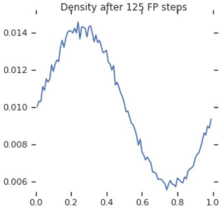
with $\tilde{f}(x) = 2\pi^2 \left[-\sum_{i=1}^d c\sin(2\pi x_i) + \sum_{i=1}^d |c\cos(2\pi x_i)|^2 \right] - 2\sum_{i=1}^d c\sin(2\pi x_i)$,

then the solution is given by $u(x) = c\sum_{i=1}^d \sin(2\pi x_i)$ and $m(x) = e^{2u(x)} / \int e^{2u}$

[Elie et al., 2020b]: Fictitious Play & DDPG (continuous spaces)



Analytical m



m Learnt by Deep RL

Flocking

Example (Flocking aversion model of [Nourian et al., 2011])

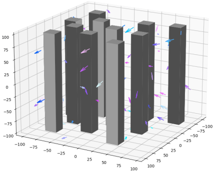
state = (position, velocity) = $(x, v) \in \mathbb{R}^{2d}$, $\begin{cases} x_{n+1} = x_n + v_n \Delta t, \\ v_{n+1} = v_n + a_n \Delta t + \epsilon_{n+1}, \end{cases}$

with running cost: $$f_\beta^{\text{flock}}(x, v, \mu) = \left\| \int_{\mathbb{R}^{2d}} \frac{(v - v')}{(1 + \|x - x'\|^2)^\beta} d\mu(x', v') \right\|^2,$$

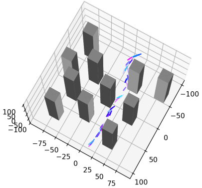
where $\beta \geq 0$, and μ is the position-velocity distribution.

[Perrin et al., 2021d]: For continuous space problems: **Deep RL**

- Deep RL (SAC) for the policy ($\approx$ control)
- Deep NN (normalizing flow) for the population distribution



Initial distribution



At convergence

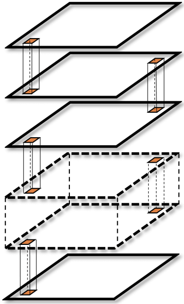
Video: `https://www.youtube.com/watch?v=TdXysW_FA3k`

Example (Crowd motion during building evacuation)

Grid world with movement to neighboring cells, and reward:

$$r(x, a, \mu) = -\eta \log(\mu(x)) + 10 \times \mathbb{1}_{floor=0}$$

Inspired by [Djehiche et al., 2017]



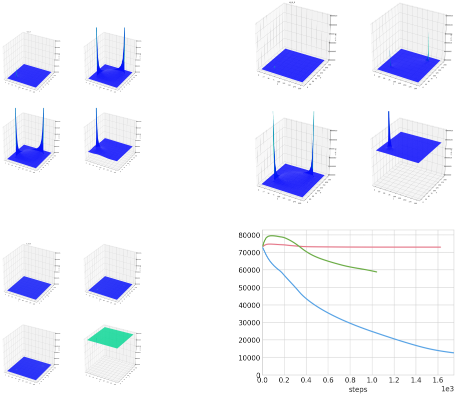
Initial distribution

Example (Crowd motion during building evacuation)

Grid world with movement to neighboring cells, and reward:

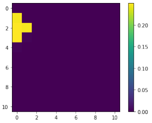
$$r(x, a, \mu) = -\eta \log(\mu(x)) + 10 \times \mathbb{1}_{floor=0}$$

Inspired by [Djehiche et al., 2017]

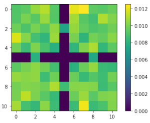


FP (red, $\alpha = 10^{-5}$), FP damped (green, $\alpha = 10^{-3}$) and OMD (blue, $\alpha = 10^{-4}$)

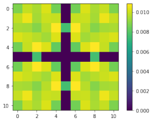
Crowd motion in 2D grid world, $r(x, a, \mu) = -\log(\mu(x))$. (See also lecture 1)



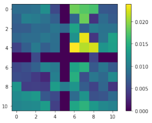
Fixed point



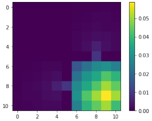
Fictitious Play



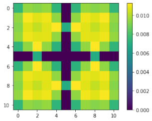
OMD



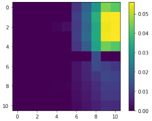
Damped Fixed Point



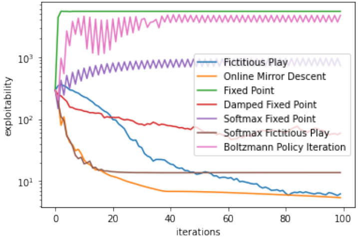
Softmax Fixed Point



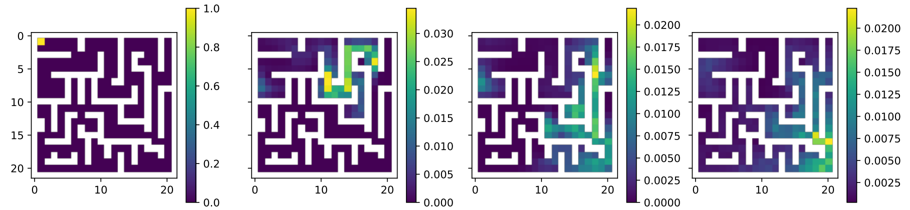
Softmax FP

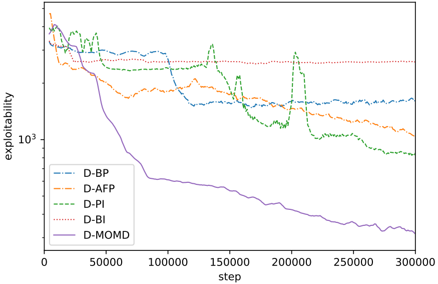


Boltzmann PI



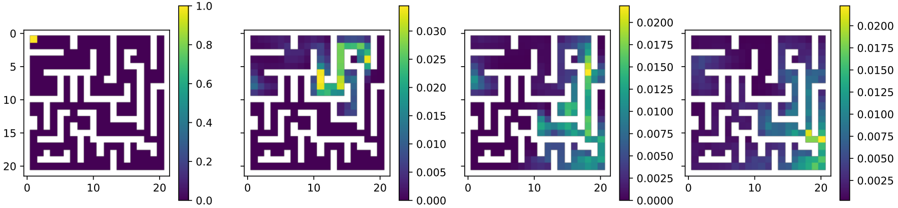
Crowd exiting a maze, with congestion effects in the reward
Deep RL combined with Online Mirror Descent & Fictitious Play

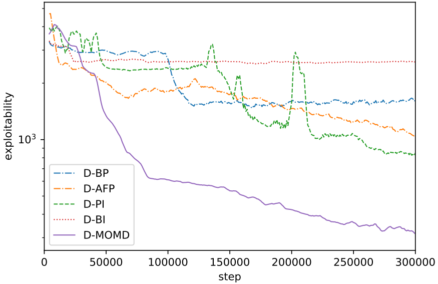




Crowd exiting a maze, with congestion effects in the reward

Deep RL combined with Online Mirror Descent & Fictitious Play





You can reproduce this experiment in OpenSpiel! (see next section)

MFControl: Fix a control α, compute induced distribution μ^{α}, update α, ...

MFGame: Fix a distribution μ, compute best response α^{μ}, update μ, ...

MFControl: Fix a control α, compute induced distribution μ^{α}, update α, ...

MFGame: Fix a distribution μ, compute best response α^{μ}, update μ, ...

Unification: update both α, μ simultaneously but at different rates $\rho^{\alpha}, \rho^{\mu}$

- $\rho^{\alpha} < \rho^{\mu} \Rightarrow \alpha$ evolves slowly $\Rightarrow$ MFControl
- $\rho^{\alpha} > \rho^{\mu} \Rightarrow \mu$ evolves slowly $\Rightarrow$ MFGame

MFControl: Fix a control α, compute induced distribution μ^α, update α, ...

MFGame: Fix a distribution μ, compute best response α^μ, update μ, ...

Unification: update both α, μ simultaneously but at different rates ρ^α, ρ^μ

- $\rho^\alpha < \rho^\mu \Rightarrow \alpha$ evolves slowly $\Rightarrow$ MFControl
- $\rho^\alpha > \rho^\mu \Rightarrow \mu$ evolves slowly $\Rightarrow$ MFGame

Implementation: Finite state space $\mathcal{X}$ and finite action space $\mathcal{A}$, stationary problem

Q-learning: Given μ, **optimal** cost-to-go when starting at x using action a

$$Q(x,a) = f(x,\mu,a) + \sum_{x' \in \mathcal{X}} p(x'|x,\mu,a) \underbrace{\min_{a'} Q(x',a')}_{=V(x')}.$$

Note: optimal control is $\hat{\alpha}_Q(x) = \mathrm{argmin}_a Q(x,a)$.

MFControl: Fix a control α, compute induced distribution μ^α, update α, ...

MFGame: Fix a distribution μ, compute best response α^μ, update μ, ...

Unification: update both α, μ simultaneously but at different rates ρ^α, ρ^μ

- $\rho^\alpha < \rho^\mu \Rightarrow \alpha$ evolves slowly $\Rightarrow$ MFControl
- $\rho^\alpha > \rho^\mu \Rightarrow \mu$ evolves slowly $\Rightarrow$ MFGame

Implementation: Finite state space $\mathcal{X}$ and finite action space $\mathcal{A}$, stationary problem

Q-learning: Given μ, **optimal** cost-to-go when starting at x using action a

$$Q(x,a) = f(x,\mu,a) + \sum_{x' \in \mathcal{X}} p(x'|x,\mu,a) \underbrace{\min_{a'} Q(x',a')}_{=V(x')} .$$

Note: optimal control is $\hat{\alpha}_Q(x) = \mathrm{argmin}_a\, Q(x,a)$.

The scheme can be written as: $\begin{cases} Q_{k+1} &= Q_k + \rho_k^Q \mathcal{T}(Q_k,\mu_k) \\ \mu_{k+1} &= \mu_k + \rho_k^\mu \mathcal{P}(Q_k,\mu_k), \end{cases}$

where $\begin{cases} \mathcal{T}(Q,\mu)(x,a) = f(x,a,\mu) + \gamma \sum_{x'} p(x'|x,a,\mu) \min_{a'} Q(x',a') - Q(x,a), \\ \mathcal{P}(Q,\mu)(x) = (\mu P^{Q,\mu})(x) - \mu(x), \qquad \text{with } P^{Q,\mu}(x,x') = p(x'|x,\hat{\alpha}_Q(x),\mu) \end{cases}$

MFControl: Fix a control α, compute induced distribution μ^α, update α, ...

MFGame: Fix a distribution μ, compute best response α^μ, update μ, ...

Unification: update both α, μ simultaneously but at different rates ρ^α, ρ^μ

- $\rho^\alpha < \rho^\mu \Rightarrow \alpha$ evolves slowly $\Rightarrow$ MFControl
- $\rho^\alpha > \rho^\mu \Rightarrow \mu$ evolves slowly $\Rightarrow$ MFGame

Implementation: Finite state space $\mathcal{X}$ and finite action space $\mathcal{A}$, stationary problem

Q-learning: Given μ, **optimal** cost-to-go when starting at x using action a

$$Q(x,a) = f(x,\mu,a) + \sum_{x' \in \mathcal{X}} p(x'|x,\mu,a) \underbrace{\min_{a'} Q(x',a')}_{=V(x')}.$$

Note: optimal control is $\hat{\alpha}_Q(x) = \operatorname{argmin}_a Q(x,a)$.

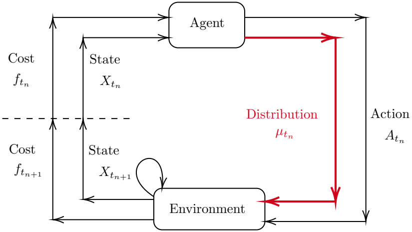
The scheme can be written as: $\begin{cases} Q_{k+1} &= Q_k + \rho^Q_k \mathcal{T}(Q_k,\mu_k) \\ \mu_{k+1} &= \mu_k + \rho^\mu_k \mathcal{P}(Q_k,\mu_k), \end{cases}$

where $\begin{cases} \mathcal{T}(Q,\mu)(x,a) = f(x,a,\mu) + \gamma \sum_{x'} p(x'|x,a,\mu) \min_{a'} Q(x',a') - Q(x,a), \\ \mathcal{P}(Q,\mu)(x) = (\mu P^{Q,\mu})(x) - \mu(x), \qquad \text{with } P^{Q,\mu}(x,x') = p(x'|x,\hat{\alpha}_Q(x),\mu) \end{cases}$

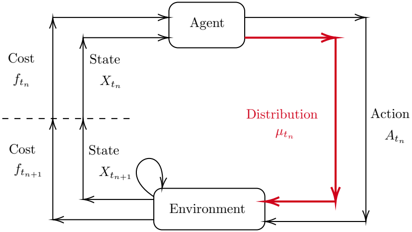
Convergence: based on Borkar's **two timescale** approach (includes sto. approx.)

Rem.: For MFG only see e.g. [Mguni et al., 2018], [Subramanian and Mahajan, 2019b]

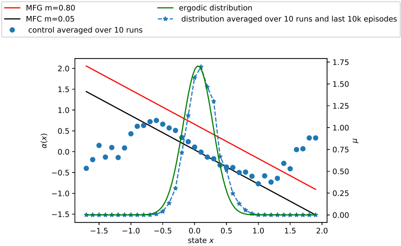
Extra difficulty: the agent needs to **estimate** the distribution



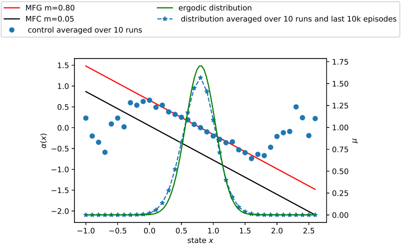
Extra difficulty: the agent needs to **estimate** the distribution



Numerical illustration: Linear-quadratic example



MFC solution ($\rho^Q < \rho^\mu$)



MFG solution ($\rho^Q > \rho^\mu$)

Comments

- Tuning properly the two learning rates is not trivial
- Proof of convergence (ongoing work with Andrea Angiuli, Jean-Pierre Fouque, and Mengrui Zhang)
- Application to other models, such as mean field control games [Angiuli et al., 2022b, Angiuli et al., 2022a]: mean field of players in a Nash equilibrium, where each agent is of mean field type (solves an MFC) $\to$ 3 time scales
- Continuous setting (ongoing work of Andrea Angiuli, Jean-Pierre Fouque, Ruimeng Hu et al.)
- RL for MFG without oracle for the distribution [Zaman et al., 2023]

Outline

OpenSpiel

- Open source framework for research in learning in games
- Main motivation: multi-agent reinforcement learning (MARL)
- Marc Lanctot (Google DeepMind) + many contributors
- Mostly in C++ and Python; APIs in Julia, ...
- Various games including zero-sum games, N-player games, imperfect information, ...
- Chess, Blackjack, Atari, Kuhn poker, Go, ...
- And also: Mean field games

OpenSpiel

Introduction to OpenSpiel:

- https://openspiel.readthedocs.io/en/latest/intro.html
- Python notebook:
  https://colab.research.google.com/github/deepmind/open_spiel/blob/master/open_spiel/colabs/OpenSpielTutorial.ipynb
- Tutorials by Marc Lanctot available online:
  https://www.youtube.com/watch?v=8NCPqtPwlFQ
- Paper [Lanctot et al., 2019]
- Two big components:
  - Games
  - Algorithms

MFG in OpenSpiel

- Julien Pérolat, Raphael Marinier, Sertan Girgin & growing number of contributors
  Théophille Cabannes, Sarah Perrin, Paul Muller, . . .

- For today, two main questions:
  - How to define a new MFG model (environment)?
  - How to define a new algorithm to learn the MFG solution?

Existing codes for MFG in OpenSpiel

- MFG models in C++: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/games/mfg`
- MFG models in Python: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/games`
  - Crowd modeling 1D illustrated in [Perrin et al., 2020]
  - Crowd modeling 2D illustrated in [Perrin et al., 2020, Geist et al., 2022]
  - Dynamic routing illustrated in [Cabannes et al., 2022]
  - Linear quadratic (1D) illustrated in [Laurière et al., 2022b]
  - Predator prey (multi-population 2D) illustrated in [Pérolat et al., 2022]

Existing codes for MFG in OpenSpiel

- MFG models in C++: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/games/mfg`
- MFG models in Python: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/games`
  - Crowd modeling 1D illustrated in [Perrin et al., 2020]
  - Crowd modeling 2D illustrated in [Perrin et al., 2020, Geist et al., 2022]
  - Dynamic routing illustrated in [Cabannes et al., 2022]
  - Linear quadratic (1D) illustrated in [Laurière et al., 2022b]
  - Predator prey (multi-population 2D) illustrated in [Pérolat et al., 2022]
- MFG algorithms in Python: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/algorithms`
  - Deep fictitious play [Laurière et al., 2022b]
  - Boltzmann policy iteration [Cui and Koeppl, 2021b]
  - Fictitious play [Perrin et al., 2020], . . .
  - Fixed point
  - Mirror descent [Pérolat et al., 2022]
  - Munchausen deep mirror descent [Laurière et al., 2022b]
  - Munchausen mirror descent

  as well as codes for policies and an evaluation metric: exploitability (nash_conv)

Existing codes for MFG in OpenSpiel

- MFG models in C++: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/games/mfg`
- MFG models in Python: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/games`
  - Crowd modeling 1D illustrated in [Perrin et al., 2020]
  - Crowd modeling 2D illustrated in [Perrin et al., 2020, Geist et al., 2022]
  - Dynamic routing illustrated in [Cabannes et al., 2022]
  - Linear quadratic (1D) illustrated in [Laurière et al., 2022b]
  - Predator prey (multi-population 2D) illustrated in [Pérolat et al., 2022]
- MFG algorithms in Python: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/algorithms`
  - Deep fictitious play [Laurière et al., 2022b]
  - Boltzmann policy iteration [Cui and Koeppl, 2021b]
  - Fictitious play [Perrin et al., 2020], . . .
  - Fixed point
  - Mirror descent [Pérolat et al., 2022]
  - Munchausen deep mirror descent [Laurière et al., 2022b]
  - Munchausen mirror descent

  as well as codes for policies and an evaluation metric: exploitability (nash_conv)
- Some examples: `https://github.com/deepmind/open_spiel/tree/master/open_spiel/python/mfg/examples`

More to come soon. Contributions are welcome!

MFG model in OpenSpiel: State

Q1. *How to define a new MFG model?*

- State of the game = all the information required to describe the current stage
- In an MFG: representative player's state and mean field state
- Evolution of the state:
  - Players play in turn
  - Every change to the state occurs through a node
  - Each node has a set of possible actions and a probability to pick each action

MFG model in OpenSpiel: State

Q1. *How to define a new MFG model?*

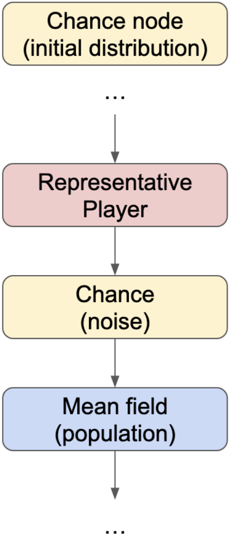
- State of the game = all the information required to describe the current stage
- In an MFG: representative player's state and mean field state
- Evolution of the state:
  - Players play in turn
  - Every change to the state occurs through a node
  - Each node has a set of possible actions and a probability to pick each action
  - So: the representative player is a node
  - the "mean field" is viewed as a node
  - and the "noise" is viewed as a node too

MFG model in OpenSpiel: State

Q1. *How to define a new MFG model?*

- State of the game = all the information required to describe the current stage
- In an MFG: representative player's state and mean field state
- Evolution of the state:
  - Players play in turn
  - Every change to the state occurs through a node
  - Each node has a set of possible actions and a probability to pick each action
  - So: the representative player is a node
  - the "mean field" is viewed as a node
  - and the "noise" is viewed as a node too
  - Time is part of the state: (t, x)
- The state evolves along a tree of possibilities

MFG model in OpenSpiel: State evolution



MFG model in OpenSpiel: State types

- Initial **chance** node:
  - actions: possible states
  - probabilities: given by the initial state distribution

MFG model in OpenSpiel: State types

- Initial **chance** node:
  - actions: possible states
  - probabilities: given by the initial state distribution
- **Player:**
  - actions: set of possible (“legal”) actions for the player
  - probabilities: given by the policy used by this player

MFG model in OpenSpiel: State types

- Initial **chance** node:
  - actions: possible states
  - probabilities: given by the initial state distribution
- **Player:**
  - actions: set of possible (“legal”) actions for the player
  - probabilities: given by the policy used by this player
- **Chance:**
  - actions: set of possible values for the noise impacting the dynamics
  - probabilities: distribution of the noise values

MFG model in OpenSpiel: State types

- Initial **chance** node:
  - actions: possible states
  - probabilities: given by the initial state distribution
- **Player:**
  - actions: set of possible (“legal”) actions for the player
  - probabilities: given by the policy used by this player
- **Chance:**
  - actions: set of possible values for the noise impacting the dynamics
  - probabilities: distribution of the noise values
- **Mean field:** no actions

MFG in OpenSpiel: Distribution

- The distribution is something specific to MFGs (compared with other games in OpenSpiel)

- Remember that time is part of the state object. Evaluating the distribution at a given state means evaluating the distribution at (t, x).

- `master/open_spiel/python/mfg/algorithms/distribution.py`
  - Computes the distribution of a policy
  - `DistributionPolicy`
    - `evaluate`: based on the logic behind nodes
    - `_one_forward_step`

- `master/open_spiel/python/mfg/distribution.py`
  - Representation of a distribution for a game
  - `Distribution`

- `master/open_spiel/python/mfg/tabular_distribution.py`
  - Tabular representation of a distribution for a game
  - `TabularDistribution`

MFG model in OpenSpiel: Example

We take a concrete example: crowd modeling in 1D with a grid world

`master/open_spiel/python/mfg/games/crowd_modelling.py`

3 main classes

- `MFGCrowdModellingGame`:
  - `__init__`: initialization
  - `new_initial_state`: generate new initial state
- `MFGCrowdModellingState`:
  - `__init__`: initialization
  - `_legal_actions`: actions that are valid
  - `chance_outcomes`: distribution over values of the noise in the dynamics
  - `_apply_action`: will be called at each node to modify the state based on the action
  - `_rewards`: representative player's reward
- `Observer`:
  - defines an observation, here basically t and x

MFG algorithms in OpenSpiel: Principles

Q2. *How to define a new algorithm?*

Simplest one: Fixed point
`master/open_spiel/python/mfg/algorithms/fixed_point.py`

A bit more involved: Fictitious play
`master/open_spiel/python/mfg/algorithms/fictitious_play.py`

- Main class `FictitiousPlay`
- Main method `iteration`
  - Compute the distribution (sequence) associated to the current policy
  - Update the policy (using fictitious play rule); this uses an auxiliary class `MergedPolicy` to mix the previous policy and the new one
- `get_policy`: returns the current policy

MFG algorithms in OpenSpiel: Reinforcement Learning

Two building blocks:

- Environment (in the sense of RL): in charge of updating the State based on the based on the Game
- Agent: block in charge of training the policy by interacting with the environment

Example of DQN (fixed distribution):

```
master/open_spiel/python/mfg/examples/mfg_dqn_jax.py
```

MFG algorithms in OpenSpiel: Reinforcement Learning

Two building blocks:

- Environment (in the sense of RL): in charge of updating the State based on the based on the Game
- Agent: block in charge of training the policy by interacting with the environment

Example of DQN (fixed distribution):

`master/open_spiel/python/mfg/examples/mfg_dqn_jax.py`

Example of DQN embedded in Fictitious Play (updating the distribution):

`master/open_spiel/python/mfg/examples/mfg_dqn_fp_jax.py`

Key steps:

- `fp.iteration(br_policy=joint_avg_policy)`: performs one iteration of fictitious play (updates the policy and the distribution)
- `distrib = distribution.DistributionPolicy(game, fp.get_policy())`: get the distribution induced by the new policy, just computed by fictitious play iteration
- `env.update_mfg_distribution(distrib)`: update the environment's distribution using the one obtained from the fictitious play iteration
- `agents[p].step(time_step)`: train the agent

Sample code

Code

Sample code to illustrate: IPython notebook

https://colab.research.google.com/drive/1HyDFqZ-qMW25sL1zyR2qYv86f_ldrm5g?usp=sharing

- MFG example in OpenSpiel

Outline

Summary (of this lecture)

- Background on RL
- RL for MFC
  - Mean Field MDP viewpoint
- RL for MFG
  - Meta-algorithm to update the mean field
  - RL algorithm to update the policy
- Open Spiel
- Survey paper: [Laurière et al., 2022a]

Summary of this course

Some References

- Introduction to Mean Field Games:

  - Pierre-Louis Lions' lectures at Collège de France (`https://www.college-de-france.fr/`)
  - Pierre Cardaliaguet's notes (2013): `https://www.ceremade.dauphine.fr/ cardaliaguet/MFG20130420.pdf`
  - Gomes, D. A., & Saúde, J. (2014). Mean field games models—a brief survey. Dynamic Games and Applications, 4, 110-154.
  - Cardaliaguet, P., & Porretta, A. (2020). An Introduction to Mean Field Game Theory. In *Mean Field Games* (pp. 1-158). Springer, Cham.
  - Carmona, Delarue, Graves, Lacker, Laurière, Malhamé & Ramanan: Lecture notes of the 2020 AMS Short Course on Mean Field Games (American Mathematical Society), organized by François Delarue
  - Achdou, Y., Cardaliaguet, P., Delarue, F., Porretta, A., & Santambrogio, F. (2021). Mean Field Games: Cetraro, Italy 2019 (Vol. 2281). Springer Nature.
  - Delarue, F. (Ed.). (2021). Mean Field Games (Vol. 78). American Mathematical Society.

- Monographs on Mean Field Games and Mean Field Control:

  - Bensoussan, A., Frehse, J., & Yam, P. (2013). *Mean field games and mean field type control theory* (Vol. 101). New York: Springer.
  - Gomes, D. A., Pimentel, E. A., & Voskanyan, V. (2016). *Regularity theory for mean-field game systems. New York: Springer.*
  - Carmona, R., & Delarue, F. (2018). *Probabilistic Theory of Mean Field Games with Applications I: Mean Field FBSDEs, Control, and Games* (Vol. 83). Springer.
  - Carmona, R., & Delarue, F. (2018). *Probabilistic Theory of Mean Field Games with Applications II: Mean Field Games with Common Noise and Master Equations* (Vol. 84). Springer.

- Surveys about numerical methods for MFGs:

  - Achdou, Y. (2013). Finite difference methods for mean field games. In *Hamilton-Jacobi equations: approximations, numerical analysis and applications* (pp. 1-47). Springer, Berlin, Heidelberg.
  - Achdou, Y., & Laurière, M. (2020). Mean Field Games and Applications: Numerical Aspects. *Mean Field Games: Cetraro, Italy 2019, 2281*, 249.
  - Laurière, M. (2021). Numerical Methods for Mean Field Games and Mean Field Type Control. Lecture notes for the AMS'20 short course. arXiv preprint arXiv:2106.06231.
  - Carmona, R., & Laurière, M. (2021). Deep Learning for Mean Field Games and Mean Field Control with Applications to Finance. arXiv preprint arXiv:2107.04568.
  - Hu, R., & Laurière, M. (2023). Recent developments in machine learning methods for stochastic control and games. arXiv preprint arXiv:2303.10257.
  - Laurière, M., Perrin, S., Geist, M., & Pietquin, O. (2022). Learning mean field games: A survey. arXiv preprint arXiv:2205.12944.

Thank you for your attention

Questions?

Feel free to reach out: `mathieu.lauriere@nyu.edu`

References I

[Almulla et al., 2017] Almulla, N., Ferreira, R., and Gomes, D. (2017).
Two numerical approaches to stationary mean-field games.
Dyn. Games Appl., 7(4):657–682.

[Anahtarci et al., 2019] Anahtarci, B., Kariksiz, C. D., and Saldi, N. (2019).
Fitted q-learning in mean-field games.
arXiv preprint arXiv:1912.13309.

[Anahtarci et al., 2020a] Anahtarci, B., Kariksiz, C. D., and Saldi, N. (2020a).
Q-learning in regularized mean-field games.

[Anahtarci et al., 2020b] Anahtarci, B., Kariksiz, C. D., and Saldi, N. (2020b).
Q-learning in regularized mean-field games.
arXiv preprint arXiv:2003.12151.

[Anahtarcı et al., 2021] Anahtarcı, B., Karıksız, C. D., and Saldi, N. (2021).
Learning in discounted-cost and average-cost mean-field games.

[Angiuli et al., 2022a] Angiuli, A., Detering, N., Fouque, J.-P., Lauriere, M., and Lin, J. (2022a).
Reinforcement learning algorithm for mixed mean field control games.
arXiv preprint arXiv:2205.02330.

[Angiuli et al., 2022b] Angiuli, A., Detering, N., Fouque, J.-P., Laurière, M., and Lin, J. (2022b).
Reinforcement learning for intra-and-inter-bank borrowing and lending mean field control game.
In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 369–376.

References II

[Angiuli et al., 2020a] Angiuli, A., Fouque, J.-P., and Laurière, M. (2020a).
Convergence of two-timescale stochastic approximation for learning MFG and MFC.
In preparation.

[Angiuli et al., 2020b] Angiuli, A., Fouque, J.-P., and Laurière, M. (2020b).
Unified reinforcement q-learning for mean field game and control problems.
arXiv preprint arXiv:2006.13912.

[Angiuli et al., 2020c] Angiuli, A., Fouque, J.-P., and Laurière, M. (2020c).
Unified reinforcement q-learning for mean field game and control problems.
arXiv preprint arXiv:2006.13912.

[Angiuli and Hu, 2021] Angiuli, A. and Hu, R. (2021).
Deep reinforcement learning for mean field games and mean field control problems in continuous spaces.
In preparation.

[Anthony et al., 2017] Anthony, T., Tian, Z., and Barber, D. (2017).
Thinking fast and slow with deep learning and tree search.
In *Proceedings of NeurIPS*.

[Bertsekas and Shreve, 1996] Bertsekas, D. P. and Shreve, S. E. (1996).
Stochastic optimal control: the discrete-time case, volume 5.
Athena Scientific.

References III

[Bowling et al., 2015] Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015).
Heads-up limit hold'em poker is solved.
Science, 347(6218).

[Brown and Sandholm, 2017] Brown, N. and Sandholm, T. (2017).
Superhuman AI for heads-up no-limit poker: Libratus beats top professionals.
Science, 360(6385).

[Brown and Sandholm, 2019] Brown, N. and Sandholm, T. (2019).
Superhuman AI for multiplayer poker.
Science, 365(6456).

[Cabannes et al., 2022] Cabannes, T., Laurière, M., Perolat, J., Marinier, R., Girgin, S., Perrin, S., Pietquin, O., Bayen, A. M., Goubault, E., and Elie, R. (2022).
Solving n-player dynamic routing games with congestion: A mean-field approach.
In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1557–1559.

[Cacace et al., 2021] Cacace, S., Camilli, F., and Goffi, A. (2021).
A policy iteration method for mean field games.
ESAIM: Control, Optimisation and Calculus of Variations, 27:85.

[Camilli and Tang, 2022] Camilli, F. and Tang, Q. (2022).
Rates of convergence for the policy iteration method for mean field games systems.
Journal of Mathematical Analysis and Applications, 512(1):126138.

[Campbell et al., 2002] Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. (2002).
Deep Blue.
Artificial intelligence, 134(1-2).

[Cardaliaguet and Hadikhanloo, 2017] Cardaliaguet, P. and Hadikhanloo, S. (2017).
Learning in mean field games: the fictitious play.
ESAIM Control Optim. Calc. Var., 23(2):569–591.

[Carmona et al., 2015] Carmona, R., Fouque, J.-P., and Sun, L.-H. (2015).
Mean field games and systemic risk.
Commun. Math. Sci., 13(4):911–933.

[Carmona et al., 2020] Carmona, R., Hamidouche, K., Laurière, M., and Tan, Z. (2020).
Policy optimization for linear-quadratic zero-sum mean-field type games.
In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1038–1043. IEEE.

[Carmona et al., 2019a] Carmona, R., Laurière, M., and Tan, Z. (2019a).
Linear-quadratic mean-field reinforcement learning: Convergence of policy gradient methods.
Preprint.

[Carmona et al., 2019b] Carmona, R., Laurière, M., and Tan, Z. (2019b).
Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning.
arXiv preprint arXiv:1910.12802.

[Carmona et al., 2019c] Carmona, R., Laurière, M., and Tan, Z. (2019c).
Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning.
To appear in Annals of Applied Probability. arXiv preprint arXiv:1910.12802.

[Chen et al., 2021] Chen, Y., Liu, J., and Khoussainov, B. (2021).
Maximum entropy inverse reinforcement learning for mean field games.
arXiv preprint arXiv:2104.14654.

[Cui and Koeppl, 2021a] Cui, K. and Koeppl, H. (2021a).
Approximately solving mean field games via entropy-regularized deep reinforcement learning.
In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1909–1917. PMLR.

[Cui and Koeppl, 2021b] Cui, K. and Koeppl, H. (2021b).
Approximately solving mean field games via entropy-regularized deep reinforcement learning.
In *International Conference on Artificial Intelligence and Statistics*, pages 1909–1917. PMLR.

[Cui et al., 2021] Cui, K., Tahir, A., Sinzger, M., and Koeppl, H. (2021).
Discrete-time mean field control with environment states.
arXiv preprint arXiv:2104.14900.

[Djehiche et al., 2017] Djehiche, B., Tcheukam, A., and Tembine, H. (2017).
A mean-field game of evacuation in multilevel building.
IEEE Transactions on Automatic Control, 62(10):5154–5169.

[Djete et al., 2019] Djete, M. F., Possamaï, D., and Tan, X. (2019).
Mckean-vlasov optimal control: the dynamic programming principle.
arXiv preprint arXiv:1907.08860.

[Elie et al., 2020a] Elie, R., Perolat, J., Laurière, M., Geist, M., and Pietquin, O. (2020a).
On the convergence of model free learning in mean field games.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7143–7150.

[Elie et al., 2020b] Elie, R., Perolat, J., Laurière, M., Geist, M., and Pietquin, O. (2020b).
On the convergence of model free learning in mean field games.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7143–7150.

[Fazel et al., 2018] Fazel, M., Ge, R., Kakade, S., and Mesbahi, M. (2018).
Global convergence of policy gradient methods for the linear quadratic regulator.
In *International Conference on Machine Learning*, pages 1467–1476. PMLR.

[Fu et al., 2019] Fu, Z., Yang, Z., Chen, Y., and Wang, Z. (2019).
Actor-critic provably finds nash equilibria of linear-quadratic mean-field games.

[Fudenberg and Levine, 2009] Fudenberg, D. and Levine, D. K. (2009).
Learning and equilibrium.
Annu. Rev. Econ., 1(1):385–420.

References VII

[Fudenberg et al., 1998] Fudenberg, D., Levine, D. K., et al. (1998).
The theory of learning in games.
MIT Press Books, 1.

[Gast and Gaujal, 2011] Gast, N. and Gaujal, B. (2011).
A mean field approach for optimization in discrete time.
Discrete Event Dynamic Systems, 21(1):63–101.

[Gast et al., 2012a] Gast, N., Gaujal, B., and Le Boudec, J.-Y. (2012a).
Mean field for markov decision processes: from discrete to continuous optimization.
IEEE Transactions on Automatic Control, 57(9):2266–2280.

[Gast et al., 2012b] Gast, N., Gaujal, B., and Le Boudec, J.-Y. (2012b).
Mean field for markov decision processes: from discrete to continuous optimization.
IEEE Transactions on Automatic Control, 57(9):2266–2280.

[Geist et al., 2022] Geist, M., Pérolat, J., Laurière, M., Elie, R., Perrin, S., Bachem, O., Munos, R., and Pietquin, O. (2022).
Concave utility reinforcement learning: The mean-field game viewpoint.
In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 489–497.

[Gu et al., 2019] Gu, H., Guo, X., Wei, X., and Xu, R. (2019).
Dynamic programming principles for mean-field controls with learning.
arXiv preprint arXiv:1911.07314.

[Gu et al., 2020a] Gu, H., Guo, X., Wei, X., and Xu, R. (2020a).
Mean-field controls with q-learning for cooperative marl: Convergence and complexity analysis.
arXiv preprint arXiv:2002.04131.

[Gu et al., 2020b] Gu, H., Guo, X., Wei, X., and Xu, R. (2020b).
Q-learning for mean-field controls.
arXiv preprint arXiv:2002.04131.

[Gu et al., 2021] Gu, H., Guo, X., Wei, X., and Xu, R. (2021).
Mean-field multi-agent reinforcement learning: A decentralized network approach.
arXiv preprint arXiv:2108.02731.

[Guo et al., 2019a] Guo, X., Hu, A., Xu, R., and Zhang, J. (2019a).
Learning mean-field games.
In *proc. of NeurIPS*.

[Guo et al., 2019b] Guo, X., Hu, A., Xu, R., and Zhang, J. (2019b).
Learning mean-field games.
Advances in Neural Information Processing Systems, 32:4966–4976.

[Guo et al., 2020] Guo, X., Hu, A., Xu, R., and Zhang, J. (2020).
A general framework for learning mean-field games.
arXiv preprint arXiv:2003.06069.

References IX

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

[Hadikhanloo, 2018] Hadikhanloo, S. (2018).
Learning in mean field games.
PhD thesis, PSL Research University.

[Hadikhanloo et al., 2021] Hadikhanloo, S., Laraki, R., Mertikopoulos, P., and Sorin, S. (2021).
Learning in nonatomic games, part i: Finite action spaces and population games.
arXiv preprint arXiv:2107.01595.

[Hadikhanloo and Silva, 2019] Hadikhanloo, S. and Silva, F. J. (2019).
Finite mean field games: fictitious play and convergence to a first order continuous mean field game.
Journal de Mathématiques Pures et Appliquées, 132:369–397.

[Huang et al., 2006] Huang, M., Malhamé, R. P., Caines, P. E., et al. (2006).
Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle.
Communications in Information & Systems, 6(3):221–252.

[Kolokoltsov and Bensoussan, 2016] Kolokoltsov, V. N. and Bensoussan, A. (2016).
Mean-field-game model for botnet defense in cyber-security.
Appl. Math. Optim., 74(3):669–692.

[Lanctot et al., 2019] Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., et al. (2019).
Openspiel: A framework for reinforcement learning in games.
arXiv preprint arXiv:1908.09453.

[Laurière, 2021] Laurière, M. (2021).
Numerical methods for mean field games and mean field type control.
arXiv preprint arXiv:2106.06231.

[Laurière et al., 2022a] Laurière, M., Perrin, S., Geist, M., and Pietquin, O. (2022a).
Learning mean field games: A survey.
arXiv preprint arXiv:2205.12944.

[Laurière et al., 2022b] Laurière, M., Perrin, S., Girgin, S., Muller, P., Jain, A., Cabannes, T., Piliouras, G., Pérolat, J., Elie, R., Pietquin, O., et al. (2022b).
Scalable deep reinforcement learning algorithms for mean field games.
In *International Conference on Machine Learning*, pages 12078–12095. PMLR.

[Laurière and Pironneau, 2016] Laurière, M. and Pironneau, O. (2016).
Dynamic programming for mean-field type control.
J. Optim. Theory Appl., 169(3):902–924.

References XI

[Laurière et al., 2023] Laurière, M., Song, J., and Tang, Q. (2023).
Policy iteration method for time-dependent mean field games systems with non-separable hamiltonians.
Applied Mathematics & Optimization, 87(2):17.

[Lillicrap et al., 2016] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016).
Continuous control with deep reinforcement learning.
In *ICLR (Poster)*.

[McAleer et al., 2020] McAleer, S., Lanier, J., Fox, R., and Baldi, P. (2020).
Pipeline PSRO: A scalable approach for finding approximate nash equilibria in large games.
In *Proceedings of NeurIPS*.

[Mguni et al., 2018] Mguni, D., Jennings, J., and de Cote, E. M. (2018).
Decentralised learning in systems with many, many strategic agents.
In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Mitchell et al., 1997] Mitchell, T. M. et al. (1997).
Machine learning.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013).
Playing atari with deep reinforcement learning.
arXiv preprint arXiv:1312.5602.

[Moravčík et al., 2017] Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017).
Deepstack: Expert-level artificial intelligence in heads-up no-limit poker.
Science, 356(6337).

[Motte and Pham, 2019a] Motte, M. and Pham, H. (2019a).
Mean-field markov decision processes with common noise and open-loop controls.
arXiv preprint arXiv:1912.07883.

[Motte and Pham, 2019b] Motte, M. and Pham, H. (2019b).
Mean-field markov decision processes with common noise and open-loop controls.
arXiv preprint arXiv:1912.07883.

[Nourian et al., 2011] Nourian, M., Caines, P. E., and Malhamé, R. P. (2011).
Mean field analysis of controlled cucker-smale type flocking: Linear analysis and perturbation equations.
IFAC Proceedings Volumes, 44(1):4471–4476.

[Pasztor et al., 2021] Pasztor, B., Bogunovic, I., and Krause, A. (2021).
Efficient model-based multi-agent mean-field reinforcement learning.
arXiv preprint arXiv:2107.04050.

[Perolat et al., 2022] Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., et al. (2022).
Mastering the game of stratego with model-free multiagent reinforcement learning.
Science, 378(6623):990–996.

[Pérolat et al., 2022] Pérolat, J., Perrin, S., Elie, R., Laurière, M., Piliouras, G., Geist, M., Tuyls, K., and Pietquin, O. (2022).
Scaling mean field games by online mirror descent.
In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1028–1037.

[Perrin et al., 2021a] Perrin, S., Laurière, M., Pérolat, J., Élie, R., Geist, M., and Pietquin, O. (2021a).
Generalization in mean field games by learning master policies.
arXiv preprint arXiv:2109.09717.

[Perrin et al., 2021b] Perrin, S., Laurière, M., Pérolat, J., Geist, M., Élie, R., and Pietquin, O. (2021b).
Mean field games flock! the reinforcement learning way.
arXiv preprint arXiv:2105.07933.

[Perrin et al., 2021c] Perrin, S., Laurière, M., Pérolat, J., Geist, M., Élie, R., and Pietquin, O. (2021c).
Mean field games flock! the reinforcement learning way.
arXiv preprint arXiv:2105.07933.

References XIV

[Perrin et al., 2021d] Perrin, S., Laurière, M., Pérolat, J., Geist, M., Élie, R., and Pietquin, O. (2021d).
Mean field games flock! the reinforcement learning way.
In *IJCAI*.

[Perrin et al., 2020] Perrin, S., Pérolat, J., Laurière, M., Geist, M., Elie, R., and Pietquin, O. (2020).
Fictitious play for mean field games: Continuous time analysis and applications.
Advances in Neural Information Processing Systems.

[Pham and Wei, 2017] Pham, H. and Wei, X. (2017).
Dynamic programming for optimal control of stochastic McKean-Vlasov dynamics.
SIAM J. Control Optim., 55(2):1069–1101.

[Schaeffer et al., 2007] Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007).
Checkers is solved.
Science, 317(5844).

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016).
Mastering the game of Go with deep neural networks and tree search.
Nature, 529(7587).

References XV

[Silver et al., 2018] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018).
A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.
Science, 632(6419).

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017).
Mastering the game of Go without human knowledge.
Nature, 550(7676).

[Subramanian and Mahajan, 2019a] Subramanian, J. and Mahajan, A. (2019a).
Reinforcement learning in stationary mean-field games.
In *AAMAS*.

[Subramanian and Mahajan, 2019b] Subramanian, J. and Mahajan, A. (2019b).
Reinforcement learning in stationary mean-field games.
In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 251–259.

[Subramanian et al., 2020a] Subramanian, S. G., Poupart, P., Taylor, M. E., and Hegde, N. (2020a).
Multi type mean field reinforcement learning.
CoRR, abs/2002.02513.

[Subramanian et al., 2020b] Subramanian, S. G., Taylor, M. E., Crowley, M., and Poupart, P. (2020b).
Partially observable mean field reinforcement learning.
CoRR, abs/2012.15791.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018).
Reinforcement learning: An introduction.
MIT press.

[Tang and Song, 2022] Tang, Q. and Song, J. (2022).
Learning optimal policies in potential mean field games: Smoothed policy iteration algorithms.
arXiv preprint arXiv:2212.04791.

[uz Zaman et al., 2022] uz Zaman, M. A., Miehling, E., and Başar, T. (2022).
Reinforcement learning for non-stationary discrete-time linear–quadratic mean-field games in multiple populations.
Dynamic Games and Applications, pages 1–47.

[uz Zaman et al., 2020] uz Zaman, M. A., Zhang, K., Miehling, E., and Bașar, T. (2020).
Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games.
In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2278–2284. IEEE.

References XVII

[Vinyals et al., 2019] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019).
Grandmaster level in StarCraft II using multi-agent reinforcement learning.
Nature, 575(7782).

[Wang et al., 2021] Wang, W., Han, J., Yang, Z., and Wang, Z. (2021).
Global convergence of policy gradient for linear-quadratic mean-field control/game in continuous time.
In *International Conference on Machine Learning*, pages 10772–10782. PMLR.

[Xie et al., 2021] Xie, Q., Yang, Z., Wang, Z., and Minca, A. (2021).
Learning while playing in mean-field games: Convergence and optimality.
In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11436–11447. PMLR.

[Yang et al., 2017] Yang, J., Ye, X., Trivedi, R., Xu, H., and Zha, H. (2017).
Deep mean field games for learning optimal behavior policy of large populations.
CoRR, abs/1711.03156.

[Yang et al., 2018] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. (2018).
Mean field multi-agent reinforcement learning.
In *Proceedings of ICML*.

References XVIII

[Yardim et al., 2022] Yardim, B., Cayci, S., Geist, M., and He, N. (2022).
Policy mirror ascent for efficient and independent learning in mean field games.
arXiv preprint arXiv:2212.14449.

[Zaman et al., 2023] Zaman, M. A. U., Koppel, A., Bhatt, S., and Basar, T. (2023).
Oracle-free reinforcement learning in mean-field games along a single sample path.
In *International Conference on Artificial Intelligence and Statistics*, pages 10178–10206. PMLR.