# Mean Field Games: Numerical Methods and Applications in Machine Learning

## Part 7: Mean Field Reinforcement Learning

Mathieu LAURIÈRE

https://mlauriere.github.io/teaching/MFG-PKU-7.pdf

Peking University
Summer School on Applied Mathematics
July 26 – August 6, 2021

# RECAP

# Outline

# From Optimal Control to MFRL

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

# 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:

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

# 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

# Reinforcement Learning – Paradigm

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

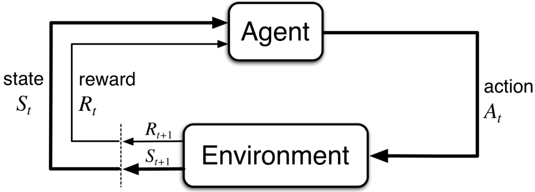
*Given the current state* $S_t$*,*
(1) *Take an action* $A_t$
(2) *Observe reward* $R_{t+1}$ *& new state* $S_{t+1}$

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

# Reinforcement Learning – Paradigm

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, Barto][1]

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

- **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
  - ...

# Reinforcement Learning – Methods

- **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: $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$, $v^*(s) = \text{argmax}_{a \in \mathcal{A}} Q^*(s, a)$

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

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

# Outline

# Problem Formulation

- **Dynamics:** discrete time

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

  - $X_n^{\alpha,\mu} \in \mathcal{X} \subseteq \mathbb{R}^d$ : 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
  - $\mu_n \in \mathcal{P}(\mathcal{X} \times \mathcal{A})$: a state-action distribution
  - $\pi_n$: a policy; randomized actions: $\alpha_n \sim \pi_n(\cdot|s_n, \mu_n)$

# Problem Formulation

- **Dynamics:** discrete time

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

  - $X_n^{\alpha,\mu} \in \mathcal{X} \subseteq \mathbb{R}^d$ : 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
  - $\mu_n \in \mathcal{P}(\mathcal{X} \times \mathcal{A})$: a state-action distribution
  - $\pi_n$: a policy; randomized actions: $\alpha_n \sim \pi_n(\cdot|s_n, \mu_n)$

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

- **Two scenarios:**
  - **Cooperative (MFC):** Find $\pi^*$ s.t.

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

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

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

# MFMDP with Common Noise & Randomization [Carmona, L., Tan'21]

- **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 [Carmona, L., Tan'21]

- **Key Remark:**

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

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

# MFMDP with Common Noise & Randomization [Carmona, L., Tan'21]

- **Key Remark:**

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

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

- **Lifted problem:** population / social planner's optimization problem:
  - $\rightarrow$ state = population distribution $\mu_n^{\pi}$
  - $\rightarrow$ value function = function of the distribution $\mu$

# MFMDP with Common Noise & Randomization [Carmona, L., Tan'21]

- **Key Remark:**

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

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

- **Lifted problem:** population / social planner's optimization problem:
  - $\rightarrow$ state = population distribution $\mu_n^\pi$
  - $\rightarrow$ value function = function of the distribution $\mu$
- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$ 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: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{X}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

# MFMDP with Common Noise & Randomization [Carmona, L., Tan'21]

- **Key Remark:**

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

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

- **Lifted problem:** population / social planner's optimization problem:
  - $\to$ state = population distribution $\mu_n^\pi$
  - $\to$ value function = function of the distribution $\mu$
- **Mean Field Markov Decision Process (MFMDP):** $(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{p}, \bar{r}, \gamma)$, where:
  - State space: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{X}\times\mathcal{U})$ 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: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{X}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$
- **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 $\bar{\mathcal{S}} \to \mathcal{P}(\bar{\mathcal{A}})$, *randomized population-strategies* $\bar{a}$

# Dynamic Programming Principle (DPP)

**Theorem:** DPP for MFMDP [Carmona, L., Tan'21][2]

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** $\bar{Q}^*$

[2] Carmona, R., Laurière, M., & Tan, Z. (2019). Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. arXiv preprint arXiv:1910.12802. (Preliminary version. Update coming soon!)

# Dynamic Programming Principle (DPP)

**Theorem:** DPP for MFMDP [Carmona, L., Tan'21][2]

Under suitable conditions,

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

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, Shreve'78]

[2] Carmona, R., Laurière, M., & Tan, Z. (2019). Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. arXiv preprint arXiv:1910.12802. (Preliminary version. Update coming soon!)

# Dynamic Programming Principle (DPP)

**Theorem:** DPP for MFMDP [Carmona, L., Tan'21][2]

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, Shreve'78]

**DPPs for MFC:**

[L., Pironneau; Pham, *et al.*; Gast *et al.*; Guo *et al.*; Possamai *et al.*;. . . ]

[2] Carmona, R., Laurière, M., & Tan, Z. (2019). Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. arXiv preprint arXiv:1910.12802. (Preliminary version. Update coming soon!)

# Dynamic Programming Principle (DPP)

**Theorem:** DPP for MFMDP [Carmona, L., Tan'21][2]

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, Shreve'78]

**DPPs for MFC:**

[L., Pironneau; Pham, *et al.*; Gast *et al.*; Guo *et al.*; Possamai *et al.*;...]

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

$\rightarrow$ well-suited for **RL**

$\rightarrow$ Mean-field Q-learning algorithm

[2] Carmona, R., Laurière, M., & Tan, Z. (2019). Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. arXiv preprint arXiv:1910.12802. (Preliminary version. Update coming soon!)

# Mean Field Learning Settings

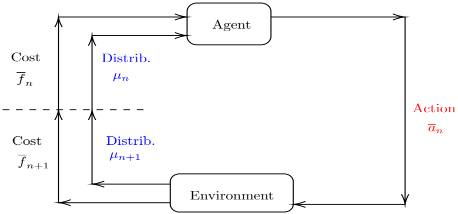
**Hierarchy** of settings:

- **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:

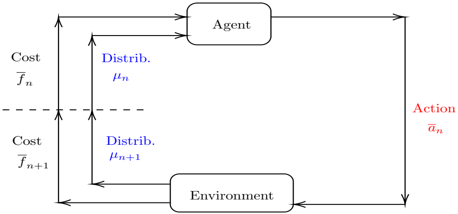
- **Setting 1: known model**: computational method based on knowledge of MFMDP
  - $(a)$ Gradient based methods
  - $(b)$ Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning



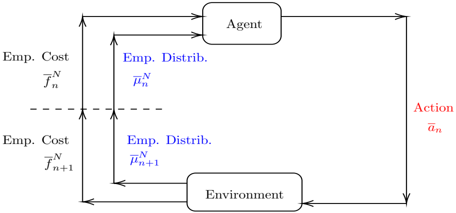
# Mean Field Learning Settings

**Hierarchy** of settings:

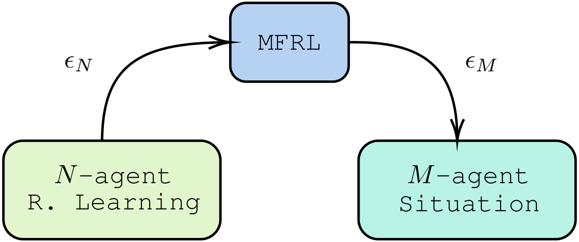
- **Setting 1: known model**: computational method based on knowledge of MFMDP
  - $(a)$ Gradient based methods
  - $(b)$ Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning




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



# Mean Field Control: Finite Population Approximation



# Outline

# Algorithm

**Idea 1:** *Make the "policy gradient" approach (see Part 5 of lecture slides) model-free*

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

- Control $\approx$ parameterized function
- Look for the optimal parameter $\theta^*$
- Perform gradient descent on the space of parameters

3

## Algorithm

**Idea 1:** *Make the "policy gradient" approach (see Part 5 of lecture slides) model-free*

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

- Control $\approx$ parameterized function
- Look for the optimal parameter $\theta^*$
- Perform gradient descent on the space of parameters

**Hierarchy** of three situations, more and more complex:

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

3

# Algorithm

**Idea 1:** *Make the "policy gradient" approach (see Part 5 of lecture slides) model-free*

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

- Control $\approx$ parameterized function
- Look for the optimal parameter $\theta^*$
- Perform gradient descent on the space of parameters

**Hierarchy** of three situations, more and more complex:

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

(2) access to a **mean field simulator**:
 $\rightarrow$ idem + gradient estimation ($0^{th}$-order opt.): $\theta^{(k+1)} = \theta^{(k)} - \eta \widetilde{\nabla} J(\theta^{(k)})$

3

# Algorithm

**Idea 1:** *Make the "policy gradient" approach (see Part 5 of lecture slides) model-free*

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

- Control $\approx$ parameterized function
- Look for the optimal parameter $\theta^*$
- Perform gradient descent on the space of parameters

**Hierarchy** of three situations, more and more complex:

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

(2) access to a **mean field simulator**:
$\rightarrow$ idem + gradient estimation ($0^{th}$-order opt.): $\theta^{(k+1)} = \theta^{(k)} - \eta \widetilde{\nabla} J(\theta^{(k)})$

(3) access to a $N$-agent **population simulator**:
$\rightarrow$ idem + error on mean $\approx$ empirical mean (LLN): $\theta^{(k+1)} = \theta^{(k)} - \eta \widetilde{\nabla}^N J(\theta^{(k)})$

3

# Algorithm

**Idea 1:** *Make the "policy gradient" approach (see Part 5 of lecture slides) model-free*

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

- Control $\approx$ parameterized function
- Look for the optimal parameter $\theta^*$
- 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, L., Tan'19][3]

In each case, convergence holds at a linear rate:

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

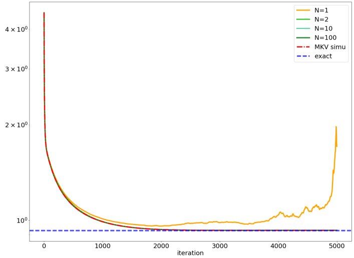
**Proof:** builds on [Fazel *et al.*'18], analysis of perturbation of Riccati equations

[3] Carmona, R., Laurière, M., & Tan, Z. (2019). Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods. arXiv preprint arXiv:1910.04295.

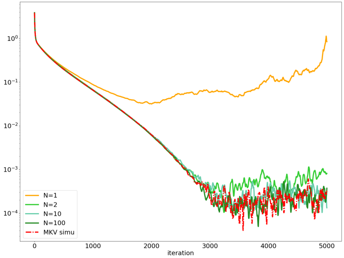
# Numerical Illustration

**Example:** Linear dynamics, quadratic costs of the type:

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




Value of the MF cost

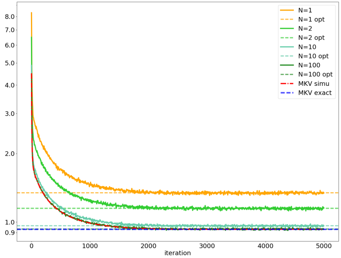



Rel. err. on MF cost

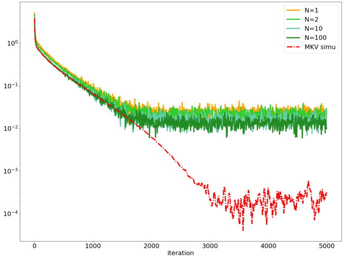
MF cost = cost in the mean field problem

**Example:** Linear dynamics, quadratic costs of the type:

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




Value of the social cost



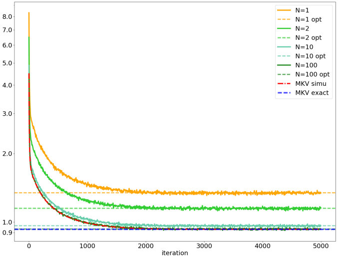

Rel. err. on social cost

Social cost = average over the $N$-agents

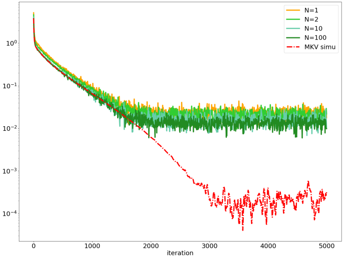
# Numerical Illustration

**Example:** Linear dynamics, quadratic costs of the type:

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




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: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$ 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: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{X}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

- **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: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - Action space: $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$ 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: $\bar{r}(\mu, \bar{a}) = -\int_{\mathcal{X}\times\mathcal{U}} f(x, a, \mu)\bar{a}(dx, da)$

- **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 $\pi$:

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

**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):

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

  - 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 $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$, $\bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$
  - deep Reinforcement Learning

# MF Q-Learning: Numerical Illustration

Cyber-security example of [Bensoussan, Kolokoltsov] (see Part 6 of slides)

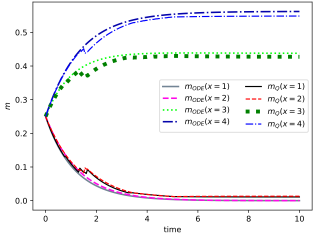
- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$

# MF Q-Learning: Numerical Illustration

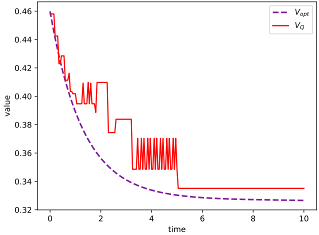
Cyber-security example of [Bensoussan, Kolokoltsov] (see Part 6 of slides)

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

**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.

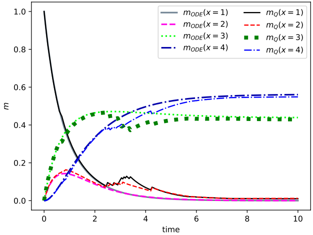
(More details in [L.'21 - AMS notes])

# MF Q-Learning: Numerical Illustration

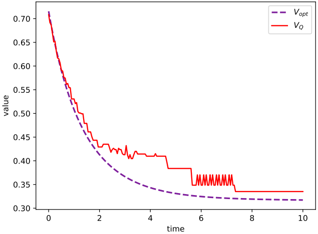
Cyber-security example of [Bensoussan, Kolokoltsov] (see Part 6 of slides)

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

**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.

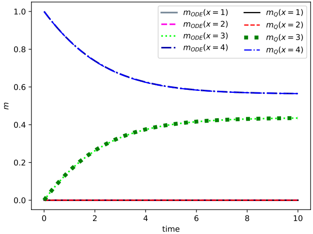
(More details in [L.'21 - AMS notes])

# MF Q-Learning: Numerical Illustration

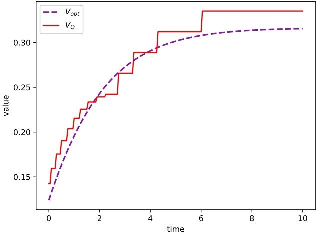
Cyber-security example of [Bensoussan, Kolokoltsov] (see Part 6 of slides)

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

**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.

(More details in [L.'21 - AMS notes])

# Summary