Mean Field Games: Numerical Methods and Applications in Machine Learning

Part 7: Mean Field Reinforcement Learning

Mathieu LAURIÈRE

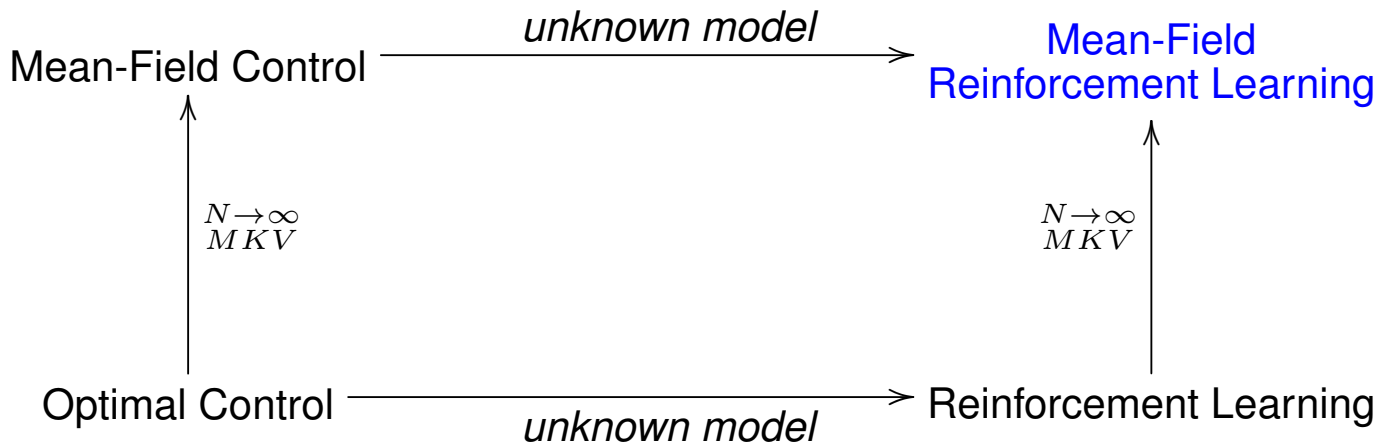
https://mlauriere.github.io/teaching/MFG-PKU-7.pdf

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

RECAP

Outline

From Optimal Control to MFRL



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.

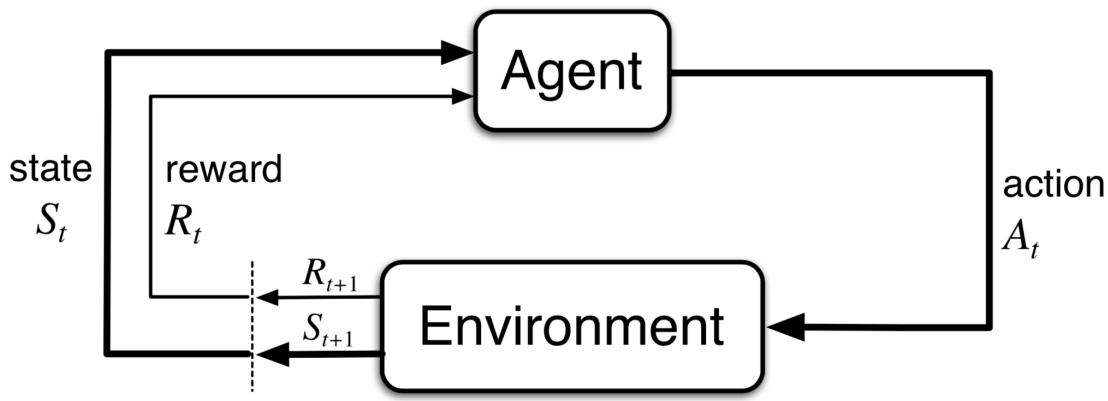
Reinforcement Learning – Paradigm

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 [SB18][1]

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

Reinforcement Learning – Methods

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

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

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
  - ...

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

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

  - 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)
  - ...

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: $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$, $v^*(s) = \text{argmax}_{a \in \mathcal{A}} Q^*(s,a)$
  - 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
  - π_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
  - π_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 π^* 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 [CLT19a])

- **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 [CLT19a])

- **Key Remark:**

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

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

MFMDP with Common Noise & Randomization (Carmona, L. & Tan [CLT19a])

- **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 μ_n^{π}
  - $\rightarrow$ value function = function of the distribution μ

MFMDP with Common Noise & Randomization (Carmona, L. & Tan [CLT19a])

- **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 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: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - 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 [CLT19a])

- **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 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: $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X})$
  - 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 $\bar{\mathcal{S}} \rightarrow \mathcal{P}(\bar{\mathcal{A}})$, *randomized population-strategies* $\bar{a}$

Dynamic Programming Principle (DPP)

Theorem: DPP for MFMDP (Carmona, L. & Tan [CLT19a])[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 [CLT19a])[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 [BS96]

[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 [CLT19a])[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 [BS96]

DPPs for MFC:

[L., Pironneau [LP16]; Pham, *et al.* [PW17]; Gast *et al.* [GGLB12]; Guo *et al.* [GGWX20]; Possamai *et al.* [DPT19];. . .]

[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 [CLT19a])[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 [BS96]

DPPs for MFC:

[L., Pironneau [LP16]; Pham, *et al.* [PW17]; Gast *et al.* [GGLB12]; Guo *et al.* [GGWX20]; Possamai *et al.* [DPT19];. . .]

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

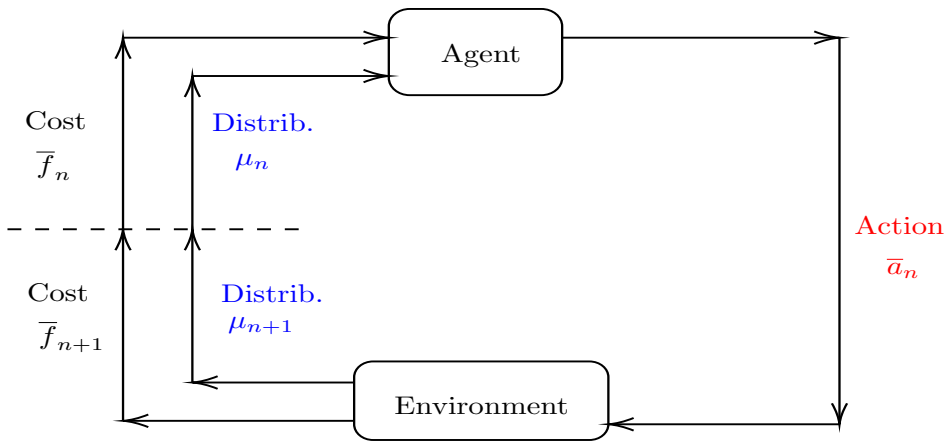
Hierarchy of settings:

- **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:

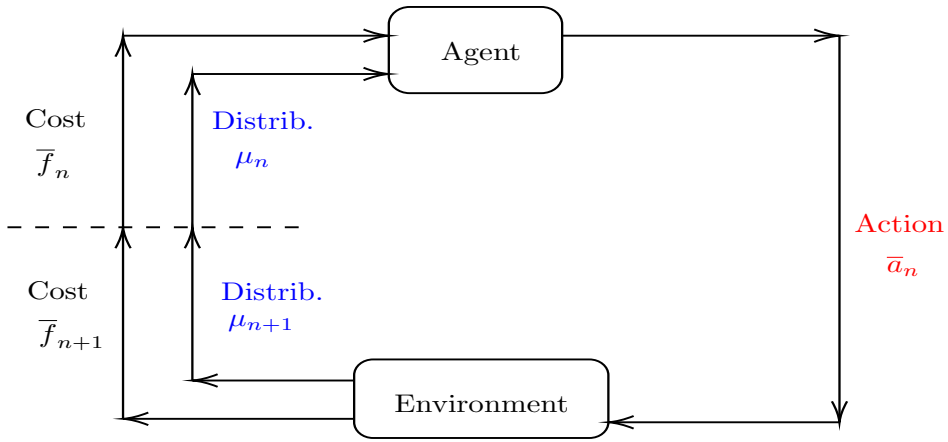
- **Setting 1: known model**: computational method based on knowledge of MFMDP
  (a) Gradient based methods
  (b) Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning



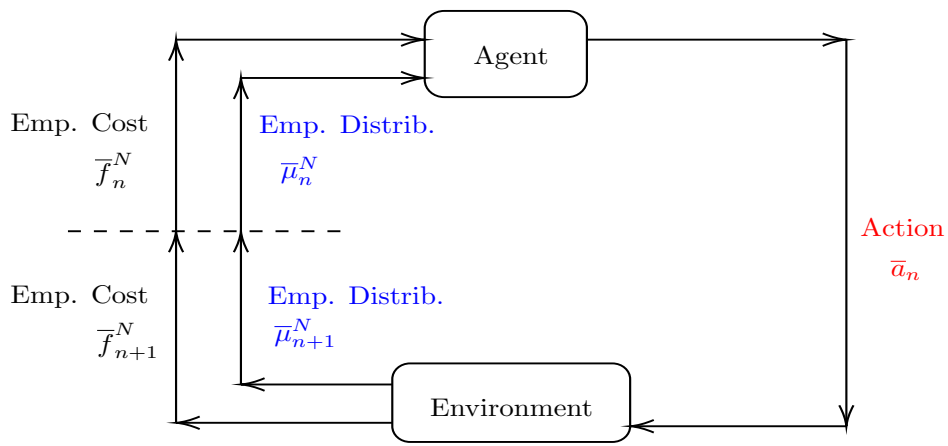
Mean Field Learning Settings

Hierarchy of settings:

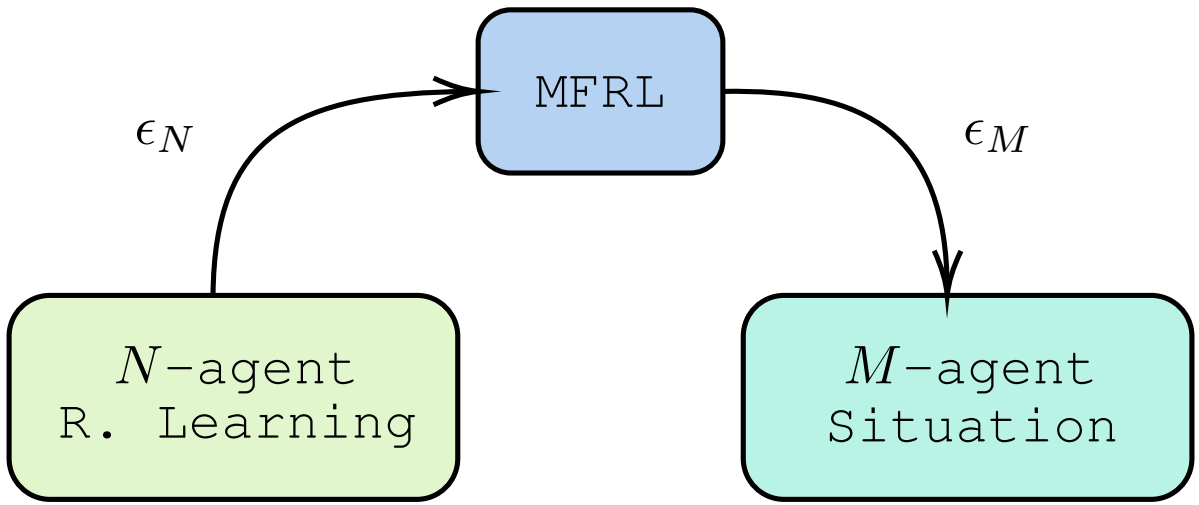
- **Setting 1: known model**: computational method based on knowledge of MFMDP
  (a) Gradient based methods
  (b) Dynamic programming based methods
- **Setting 2:** unknown model but **samples from MFMDP**: MF learning



- **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 θ^*
- 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 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**: $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla J(\theta^{(k)})$

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 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**: $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla J(\theta^{(k)})$

(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 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**: $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla J(\theta^{(k)})$

(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 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, L. & Tan [CLT19b])[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 $J(\theta^{(\mathtt{k})}) - J(\theta^*) < \epsilon$.

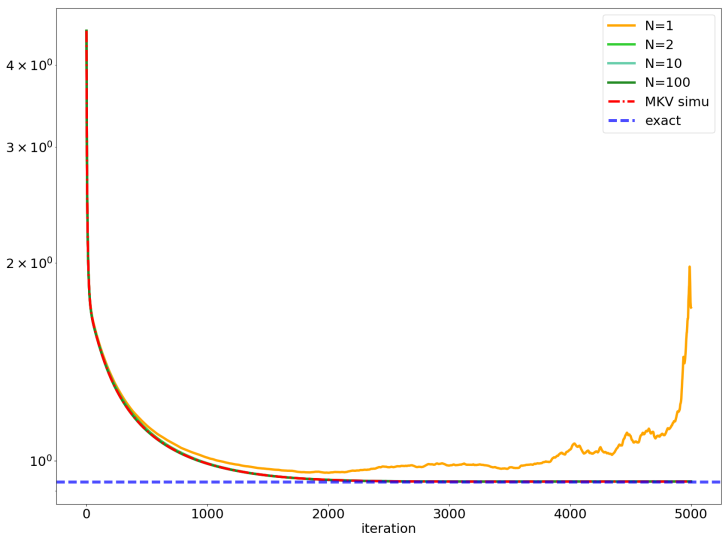
Proof: builds on Fazel *et al.* [FGKM18], analysis of perturbation of Riccati equations

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

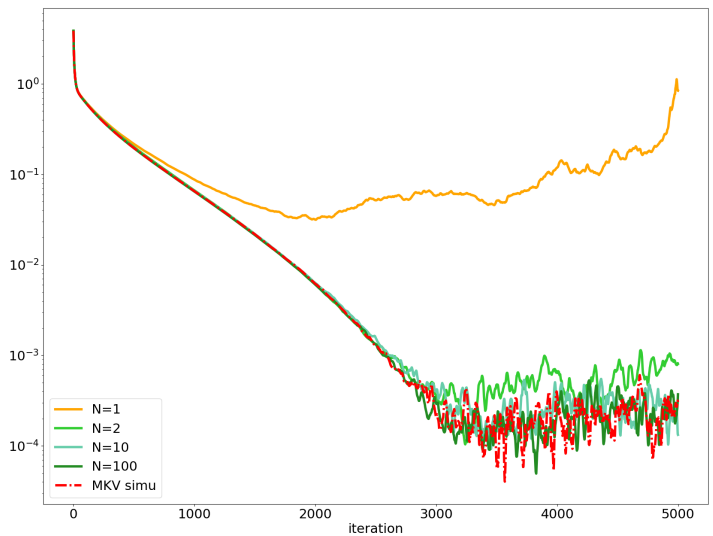
Numerical Illustration

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



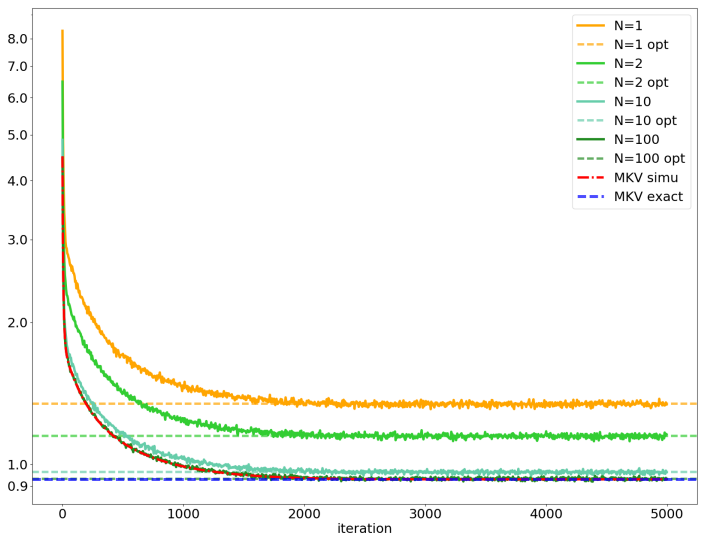
Rel. err. on MF cost

MF cost = cost in the mean field problem

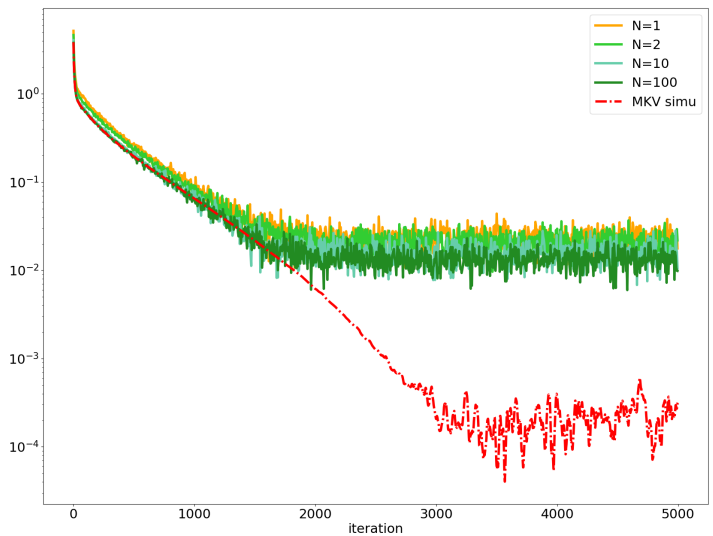
Numerical Illustration

Example: Linear dynamics, quadratic costs of the type:

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



Value of the social cost



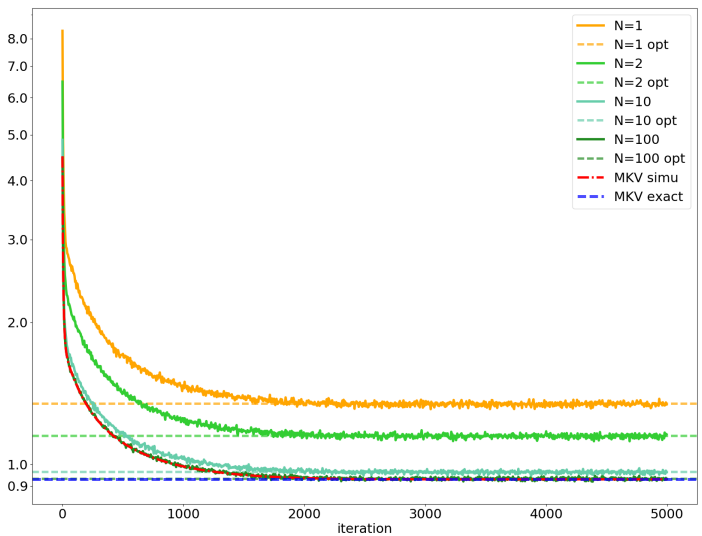
Rel. err. on social cost

Social cost = average over the N-agents

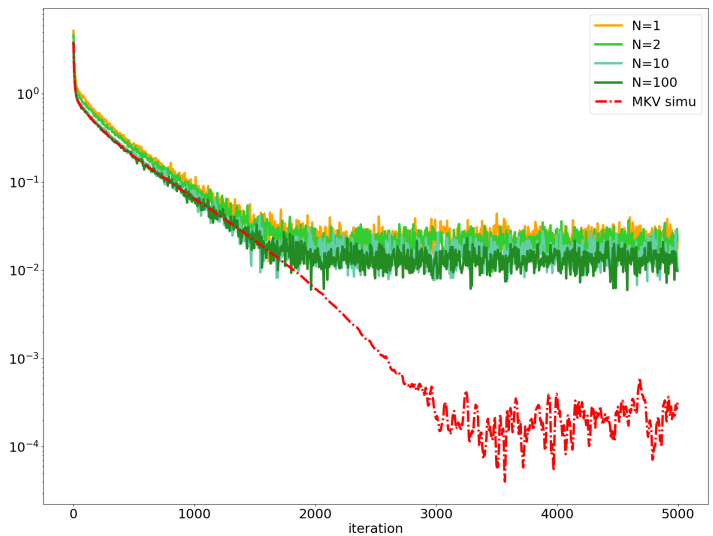
Numerical Illustration

Example: Linear dynamics, quadratic costs of the type:

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



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 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):

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

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

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

- **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 Kolokoltsov, Bensoussan [KB16] (see Part 6 of slides)

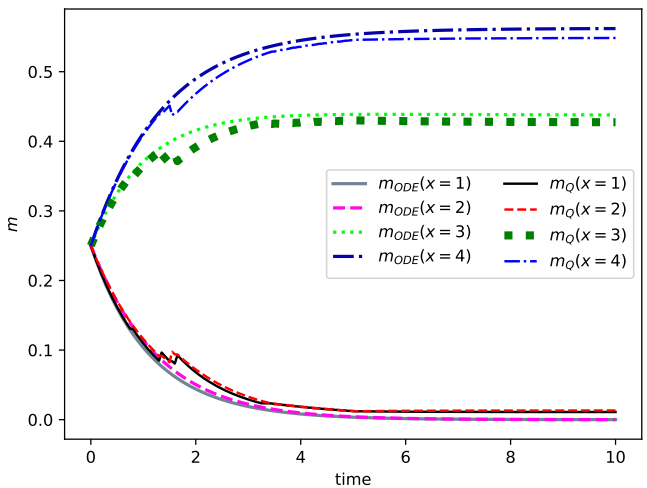
- MFC viewpoint, MF Q-learning
- pure (population and individual) strategies
- discretization of $\bar{\mathcal{S}} = \mathcal{P}(\mathcal{X}), \bar{\mathcal{A}} = \mathcal{P}(\mathcal{X} \times \mathcal{U})$

MF Q-Learning: Numerical Illustration

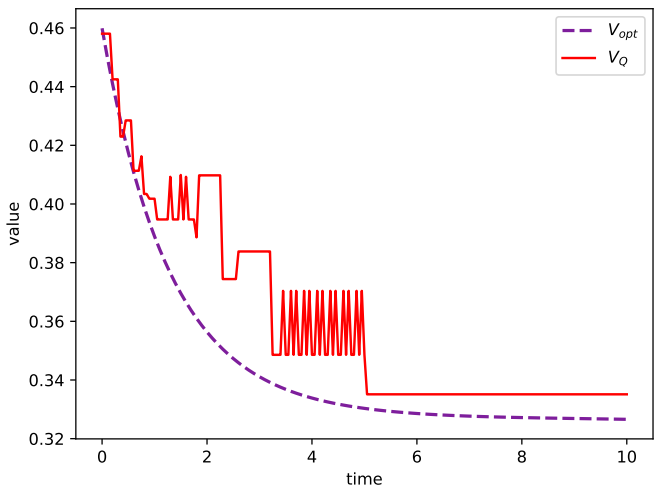
Cyber-security example of Kolokoltsov, Bensoussan [KB16] (see Part 6 of slides)

- 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.

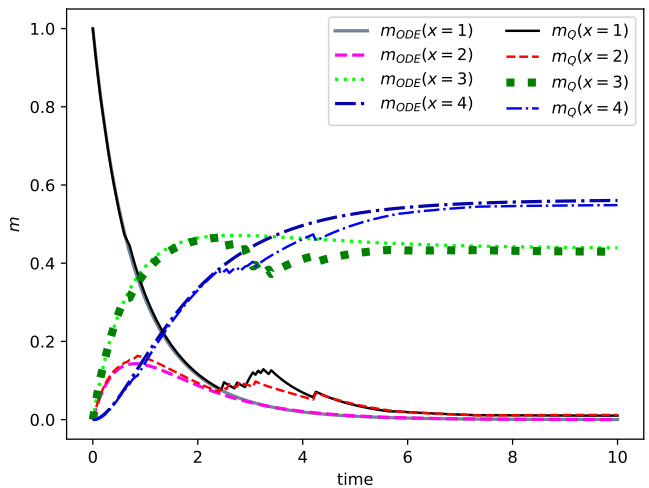
(More details in L.'21 [Lau21])

MF Q-Learning: Numerical Illustration

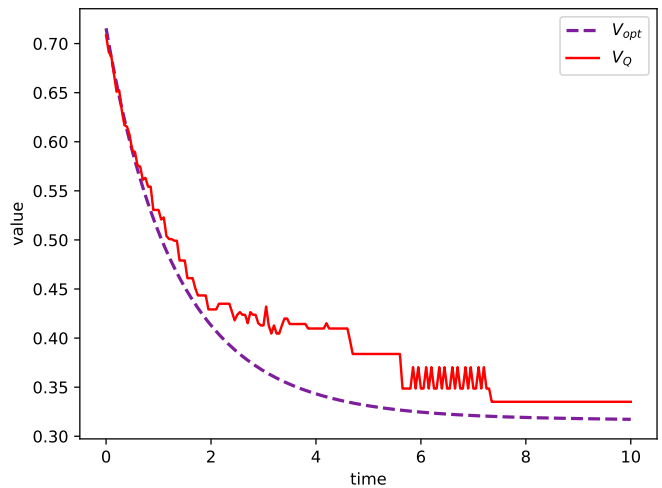
Cyber-security example of Kolokoltsov, Bensoussan [KB16] (see Part 6 of slides)

- 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 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.

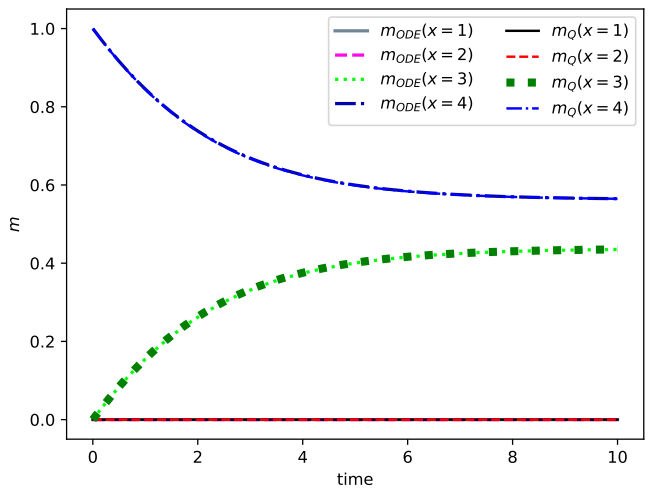
(More details in L.'21 [Lau21])

MF Q-Learning: Numerical Illustration

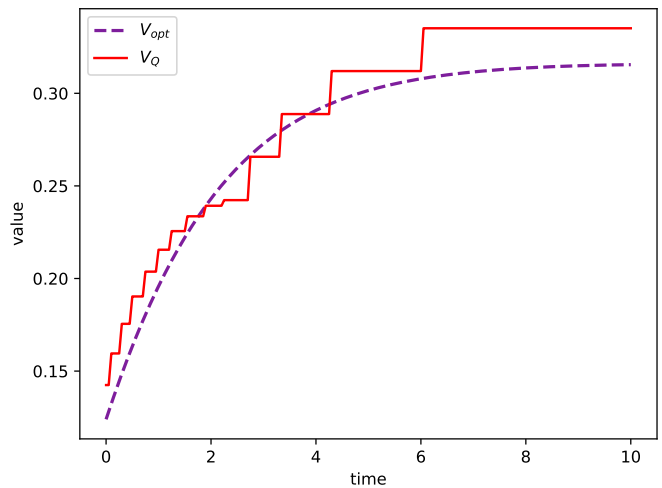
Cyber-security example of Kolokoltsov, Bensoussan [KB16] (see Part 6 of slides)

- 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 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 [Lau21])

Summary

References I

[BS96] Dimitri P Bertsekas and Steven E Shreve, *Stochastic optimal control: the discrete-time case*, vol. 5, Athena Scientific, 1996.

[CLT19a] René Carmona, Mathieu Laurière, and Zongjun Tan, *Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning*, arXiv preprint arXiv:1910.12802 (2019).

[CLT19b] _____, *Model-free mean-field reinforcement learning: Mean-field mdp and mean-field q-learning*, Preprint, October 2019.

[DPT19] Mao Fabrice Djete, Dylan Possamaï, and Xiaolu Tan, *Mckean-vlasov optimal control: the dynamic programming principle*, arXiv preprint arXiv:1907.08860 (2019).

[FGKM18] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi, *Global convergence of policy gradient methods for the linear quadratic regulator*, International Conference on Machine Learning, PMLR, 2018, pp. 1467–1476.

[GGLB12] Nicolas Gast, Bruno Gaujal, and Jean-Yves Le Boudec, *Mean field for markov decision processes: from discrete to continuous optimization*, IEEE Transactions on Automatic Control **57** (2012), no. 9, 2266–2280.

[GGWX20] Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu, *Q-learning for mean-field controls*, arXiv preprint arXiv:2002.04131 (2020).

References II

[KB16] Vassili N. Kolokoltsov and Alain Bensoussan, *Mean-field-game model for botnet defense in cyber-security*, Appl. Math. Optim. **74** (2016), no. 3, 669–692. MR 3575619

[Lau21] Mathieu Laurière, *Numerical methods for mean field games and mean field type control*, arXiv preprint arXiv:2106.06231 (2021).

[LP16] Mathieu Laurière and Olivier Pironneau, *Dynamic programming for mean-field type control*, J. Optim. Theory Appl. **169** (2016), no. 3, 902–924. MR 3501391

[PW17] Huyên Pham and Xiaoli Wei, *Dynamic programming for optimal control of stochastic McKean-Vlasov dynamics*, SIAM J. Control Optim. **55** (2017), no. 2, 1069–1101. MR 3631380

[SB18] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

Unless otherwise specified, the images are from `https://unsplash.com`