Mean Field Games: Numerical Methods and Applications in Machine Learning

Part 8: Learning in MFGs

Mathieu LAURIÈRE

https://mlauriere.github.io/teaching/MFG-PKU-8.pdf

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

RECAP

Outline

Warning

⚠ Terminology **“learning”**:

1

2

Warning

⚠ Terminology **"learning"**:

- Game theory, economics, . . . :
  [Fudenberg, Levine][1]: *"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"*

---

[1] Fudenberg, D., & Levine, D. K. (2009). Learning and equilibrium. *Annu. Rev. Econ.*, 1(1), 385-420.
[2]

Warning

⚠ Terminology **"learning"**:

- Game theory, economics, . . . :
  [Fudenberg, Levine][1]: *"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"*

- Machine learning, RL, . . . :
  [Mitchell][2]: *"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."*

[1] Fudenberg, D., & Levine, D. K. (2009). Learning and equilibrium. *Annu. Rev. Econ.*, 1(1), 385-420.
[2] Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill. ISBN: 978-0-07-042807-2

Learning/optimization methods:

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

3

4

5

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, Levine][3]
- Non-atomic games. see e.g. [Hadikhanloo *et al.*'21][4]
- Mean Field Games, see e.g. [Hadikhanloo'18][5]

[3] Fudenberg, D., & Levine, D. (1998). The Theory of Learning in Games. *The MIT Press*.

[4] Hadikhanloo, S., Laraki, R., Mertikopoulos, P., & Sorin, S. (2021). Learning in nonatomic games, Part I: Finite action spaces and population games. arXiv preprint arXiv:2107.01595.

[5] Hadikhanloo, S. (2018). *Learning in Mean Field Games* (Doctoral dissertation, Université Paris sciences et lettres).

Generic structure: repeated game (iterations)

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

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

MFG Setup

Generic Mean Field model: for a typical infinitesimal agent

- **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: $\alpha_n \sim \pi_n(\cdot|s_n, \mu_n)$

MFG Setup

Generic Mean Field model: for a typical infinitesimal agent

- **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: $\alpha_n \sim \pi_n(\cdot|s_n, \mu_n)$

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

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

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

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

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

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:

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

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:

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

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**?*

→ First type of "Learning": meta-algorithm / outside loop

→ Second type of "Learning": agent's viewpoint / inner loop

Banach-Picard Fixed Point Iterations

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

- 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

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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

⚠ First assumption is hard to check! Can be relaxed with entropy regularization

Banach-Picard Fixed Point Iterations

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

- 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

⚠ First assumption is hard to check! Can be relaxed with entropy regularization

Remark: version with damping/mixing/smoothing

Banach-Picard Fixed Point Iterations

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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

⚠ First assumption is hard to check! Can be relaxed with entropy regularization

Remark: version with damping/mixing/smoothing

See e.g., [Caines *et al.*; Guo *et al.*; Anahtarci *et al.*; . . .]

Fictitious Play

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- Update agent's policy: $\pi^{(\mathtt{k}+1)} \in \mathrm{BR}(\overline{\mu}^{(\mathtt{k})})$
- 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: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

Example: crowd aversion

Fictitious Play

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

Example: crowd aversion

Consequence:

$$0 \geq \left[J(\pi; \mu) - J(\pi; \mu')\right] - \left[J(\pi'; \mu) - J(\pi'; \mu')\right] =: \mathcal{M}(\pi, \mu, \pi', \mu')$$

If $(\hat{\mu}, \hat{\pi})$ and $(\hat{\mu}', \hat{\pi}')$ are two Nash equilibria,

$$\begin{aligned}\mathcal{M}(\hat{\pi}, \hat{\mu}, \hat{\pi}', \hat{\mu}') &= \left[J(\hat{\pi}; \hat{\mu}) - J(\hat{\pi}'; \hat{\mu})\right] + \left[J(\hat{\pi}'; \hat{\mu}') - J(\hat{\pi}; \hat{\mu}')\right] \\ &\geq \mathcal{E}(\hat{\pi}'; \hat{\mu}) + \mathcal{E}(\hat{\pi}; \hat{\mu}') \geq 0\end{aligned}$$

where $\mathcal{E}$ denotes the **exploitability** of π facing μ: $\mathcal{E}(\pi; \mu) = \sup J(\cdot; \mu) - J(\pi; \mu) \geq 0$

Fictitious Play

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

- 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: $\langle \tilde{r}(x, \mu) - \tilde{r}(x, \mu'), \mu - \mu' \rangle \leq 0$

Example: crowd aversion

Consequence:

$$0 \geq \Big[J(\pi; \mu) - J(\pi; \mu')\Big] - \Big[J(\pi'; \mu) - J(\pi'; \mu')\Big] =: \mathcal{M}(\pi, \mu, \pi', \mu')$$

If $(\hat{\mu}, \hat{\pi})$ and $(\hat{\mu}', \hat{\pi}')$ are two Nash equilibria,

$$\begin{aligned}\mathcal{M}(\hat{\pi}, \hat{\mu}, \hat{\pi}', \hat{\mu}') &= \Big[J(\hat{\pi}; \hat{\mu}) - J(\hat{\pi}'; \hat{\mu})\Big] + \Big[J(\hat{\pi}'; \hat{\mu}') - J(\hat{\pi}; \hat{\mu}')\Big] \\ &\geq \mathcal{E}(\hat{\pi}'; \hat{\mu}) + \mathcal{E}(\hat{\pi}; \hat{\mu}') \geq 0\end{aligned}$$

where $\mathcal{E}$ denotes the **exploitability** of π facing μ: $\mathcal{E}(\pi; \mu) = \sup J(\cdot; \mu) - J(\pi; \mu) \geq 0$

See e.g., [Cardaliaguet & Hadikhanloo; Elie *et al.*; Perrin *et al.*; Geist *et al.*; ...]

Banach-Picard Fixed Point Iterations – with Q-function

Reminder:

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

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

Banach-Picard Fixed Point Iterations – with Q-function

Reminder:
Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

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

Or, using a Q-function defined as:

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

Banach-Picard Fixed Point Iterations – with Q-function

Reminder:

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

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

Or, using a Q-function defined as:

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

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

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

Banach-Picard Fixed Point Iterations – with Q-function

Reminder:

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

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

Or, using a Q-function defined as:

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

Method: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

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

Online Mirror Descent

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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: For $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K}$:

- 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: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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

Online Mirror Descent

Method: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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: For $\mathtt{k} = 0, 1, 2, \ldots, \mathtt{K}$:

- 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; Pérolat *et al.*; Geist *et al.*; ...]

Outline

Learning in MFGs

Generic structure: repeated game (iterations)

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

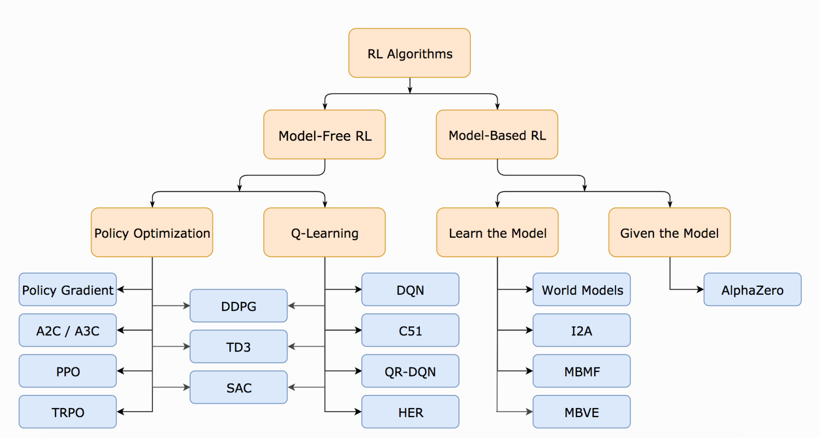
*Where is there **learning**?*

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

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

Outline

RL Taxonomy



Source: [OpenAI Spinning Up][6]

[6] 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.*'13][7]

[7] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning.

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.*'16][8]

[8] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. (2016). Continuous control with deep reinforcement learning. ICLR 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.*'18][9]

[9] Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. ICML 2018.

Population Distribution Approximation

What about the population behavior μ*?*

Population Distribution Approximation

What about the population behavior μ?

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

Outline

Systemic Risk

Revisiting: Systemic risk model of [Carmona, Fouque, Sun]

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

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

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

Systemic Risk

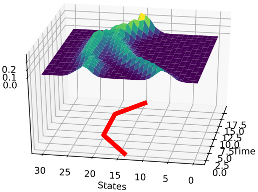
Revisiting: Systemic risk model of [Carmona, Fouque, Sun]

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

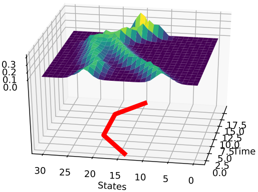
Subj. to: $$X_{n+1} = X_n + [K(m_n - X_n) + a_n] + \epsilon_{n+1} + \epsilon^0_{n+1}$$

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

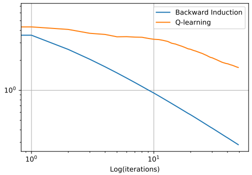
[Perrin *et al.*, NeurIPS'20]: Fictitious Play with Backward Induction or tabular Q-learning



Exact solution



Fictitious Play & RL



Exploitability

Crowd Aversion

Revisiting: Crowd aversion model of [Alumulla, Ferreira, Gomes]
MFG on $\mathbb{T}$,

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

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 $m(x) = e^{2u(x)} / \int e^{2u}$

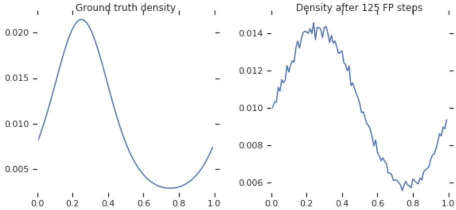
Crowd Aversion

Revisiting: Crowd aversion model of [Alumulla, Ferreira, Gomes]
MFG on $\mathbb{T}$,

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

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.*, AAAI'20]: Fictitious Play & DDPG (continuous spaces)



Analytical m $\qquad$ m Learnt by Deep RL

Flocking

Revisiting: Flocking aversion model of [Nourian, Caines, Malhamé]
[Perrin *et al.*, IJCAI'21]: For continuous space problems: **Deep RL**

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

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

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

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

Flocking

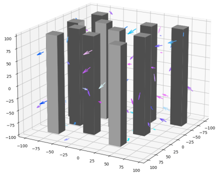
Revisiting: Flocking aversion model of [Nourian, Caines, Malhamé]
[Perrin *et al.*, IJCAI'21]: For continuous space problems: **Deep RL**

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

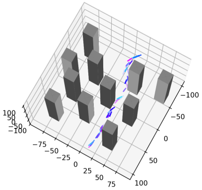
state = (position, velocity) = $(x, v) \in \mathbb{R}^{2d}$, $\begin{cases} x_{n+1} = x_n + v_n \Delta t, \\ v_{n+1} = v_n + a_n \Delta t + \epsilon_{n+1}, \end{cases}$

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.



Initial distribution



At convergence

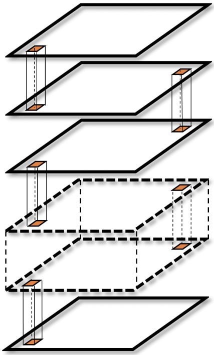
Video: `https://www.youtube.com/watch?v=TdXysW_FA3k`

Building Evacuation

A model for crowd motion during building evacuation:

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

[Pérolat *et al.*'21]: OMD (no RL)



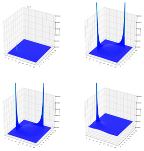
Initial distribution

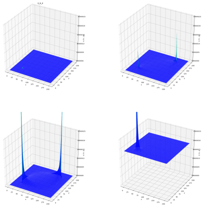
Building Evacuation

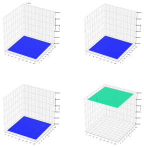
A model for crowd motion during building evacuation:

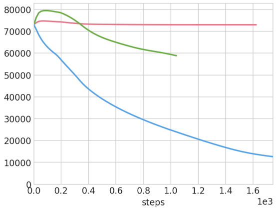
$$r(x, a, \mu) = -\eta \log(\mu(x)) + 10 \times \mathbb{1}_{floor=0}$$

[Pérolat *et al.*'21]: OMD (no RL)









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

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

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

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

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

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

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

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

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

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

- $\rho^v < \rho^\mu \Rightarrow v$ evolves slowly $\Rightarrow$ MFControl
- $\rho^v > \rho^\mu \Rightarrow \mu$ 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 $\hat{v}_Q(x) = \operatorname{argmin}_a Q(x,a)$.

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

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

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

- $\rho^v < \rho^\mu \Rightarrow v$ evolves slowly $\Rightarrow$ MFControl
- $\rho^v > \rho^\mu \Rightarrow \mu$ 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 $\hat{v}_Q(x) = \operatorname{argmin}_a Q(x,a)$.

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

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

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

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

- $\rho^v < \rho^\mu \Rightarrow v$ evolves slowly $\Rightarrow$ MFControl
- $\rho^v > \rho^\mu \Rightarrow \mu$ 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 $\hat{v}_Q(x) = \operatorname{argmin}_a Q(x,a)$.

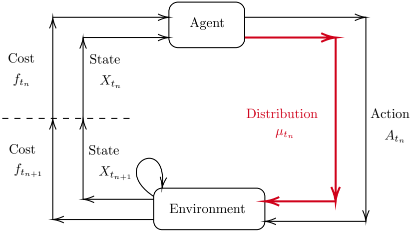
The scheme can be written as: $\begin{cases} Q_{k+1} &= Q_k + \rho_k^Q \, \mathcal{T}(Q_k, \mu_k) \\ \mu_{k+1} &= \mu_k + \rho_k^\mu \, \mathcal{P}(Q_k, \mu_k), \end{cases}$

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

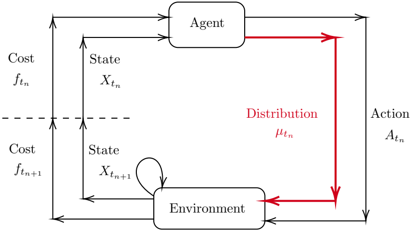
Convergence: based on Borkar's **two timescale** approach (includes sto. approx.)

Rem.: For MFG (only) see e.g. [Mguni *et al.*'18, Subrahmanian *et al.*'19]

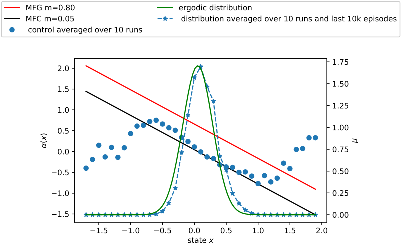
Restricted environment: the agent needs to **estimate** the distribution



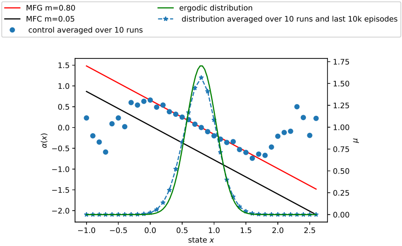
Restricted environment: the agent needs to **estimate** the distribution



Numerical illustration: Linear-quadratic example



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



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

Summary