# Présentation finale du PFE

Apprentissage par renforcement multi-agents

David Albert

INSA Rouen

21 février 2020

# Apprentissage automatique $\approx$ estimer une loi de probabilité

- Objectif : Construire $P_{model}$ proche de $P_{reel}$

# Apprentissage automatique $\approx$ estimer une loi de probabilité

- Objectif : Construire $P_{model}$ proche de $P_{reel}$
- Supervisé:

$$P\left(\mathbf{y} = \text{cat} \middle| \text{  }\right)$$

# Apprentissage automatique $\approx$ estimer une loi de probabilité

- Objectif : Construire $P_{model}$ proche de $P_{reel}$
- Supervisé:

$$P\left(\mathbf{y} = \text{cat} \middle| \text{  }\right)$$

- Non-supervisé:

 $\sim P(\mathbf{x})$

# Apprentissage automatique $\approx$ estimer une loi de probabilité

- Objectif : Construire $P_{model}$ proche de $P_{reel}$
- Supervisé:

$$P\left(\mathbf{y} = \text{cat} \middle| \text{  }\right)$$

- Non-supervisé:

 $\sim P(\mathbf{x})$

- Séquentiel:

$$P\left(X_{t+1} = \text{"}pluie\text{"} \mid X_t = \text{"}soleil\text{"}\right)$$

# Apprentissage automatique $\approx$ estimer une loi de probabilité

- Objectif : Construire $P_{model}$ proche de $P_{reel}$
- Supervisé:

$$P\left(\mathbf{y} = \text{cat} \,\middle|\, \text{  }\right)$$

- Non-supervisé:

 $\sim P(\mathbf{x})$

- Séquentiel:

$$P\left(X_{t+1} = \text{"}pluie\text{"} \mid X_t = \text{"}soleil\text{"}\right)$$

- Renforcement:

$$P\left(X_{t+1} = \text{"}Chomage \nearrow\text{"}, R_{t+1} = -100 \mid X_t = \text{"}Chomage \searrow\text{"}, A_t = \text{"}Reforme\text{"}\right)$$

et $\quad P\left(A_t = \text{"}Reforme\text{"} \mid X_t = \text{"}Chomage \searrow\text{"}\right)$

# Apprentissage par renforcement

- Etude d'un système composé de 2 éléments:
  - L'environnement (Le système économique français)
    $\rightarrow$ $p(s_t, r_t, o_t \mid s_1, r_1, o_1, a_1, s_2, \ldots, s_{t-1}, r_{t-1}, o_{t-1}, a_{t-1})$
  - L'agent (M. Macron)
    $\rightarrow$ $p(a_t \mid r_1, o_1, a_1, \ldots, a_t, r_t, o_t)$
- Planification vs Apprentissage
- Trois approches d'apprentissage:
  1. Estimation du modèle de transition $\mathcal{T}$ + planification
  2. Estimation de la fonction de valeur $q_\pi$
  3. Optimisation direct de la politique d'action $\pi$

Problème: Prise de décision repose sur l'intégralité du passé.

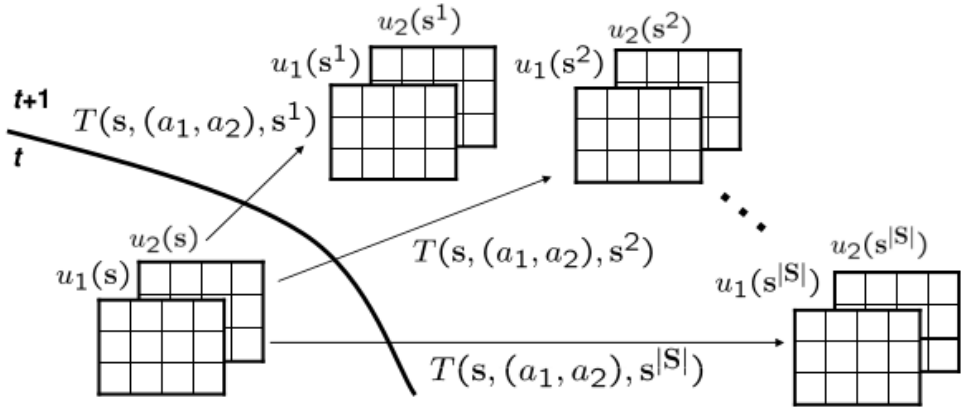
Solution: Utiliser un modèle simplifié d'évolution (MDP).

## Definition

Les **Processus de décision de Markov** est un modèle d'apprentissage par renforcement qui suit la propriété de Markov ("le futur ne dépend que tu présent").

**Jeux stochastiques (SG)** peuvent être vus comme :

1. extension des MDPs pour $N$ agents (RL)
2. jeux matriciels à étapes (Théorie des Jeux)




Types de jeux :

- Coopératifs / **Non coopératifs**
- **Forme normale** / Forme extensive

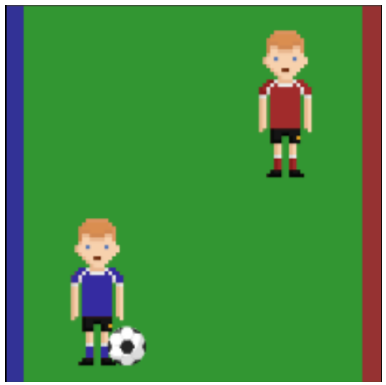
Agents rationnels:

- exploite défaut des adversaires
- équilibre de Nash

# Minimax-Q

Q-learning pour jeux à deux joueurs compétitifs.

- construit $Q_{s,a,o}$ pour chaque joueur
- $Q_{s,.,.}$ jeu matriciel



$Q_1(s)$ et $Q_2(s)$

| $A_1 \setminus A_2$ | *Front* | *Back* | *Left* | *Right* |
|---|---|---|---|---|
| *Front* | 10<br>-10 | 10<br>-10 | 6<br>-6 | 10<br>-10 |
| *Back* | 8<br>-8 | 7<br>-7 | 5<br>-5 | 7<br>-7 |
| *Left* | 3<br>-3 | 5<br>-5 | 7<br>-7 | 5<br>-5 |
| *Right* | 8<br>-8 | 7<br>-7 | 5<br>-5 | 7<br>-7 |

Mise à jour:

$$Q_i(s,a,o) = Q_i(s,a,o) + \alpha[r_t + \gamma \max_{\mathbf{a}} \min_{\mathbf{o}} \mathbf{Q_i}(\mathbf{s'},\mathbf{a},\mathbf{o}) - Q_i(s,a,o)]$$

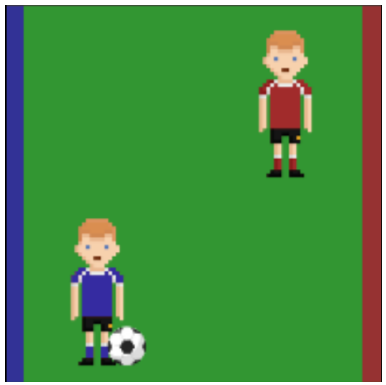
Choix de la politique:

$$\pi_i(s) = arg \max_a \min_o Q_i(s,a,o)$$

# Minimax-Q

Q-learning pour jeux à deux joueurs compétitifs.

- construit $Q_{s,a,o}$ pour chaque joueur
- $Q_{s,.,.}$ jeu matriciel



$Q_1(s)$ et $Q_2(s)$

| $A_1 \setminus A_2$ | *Front* | *Back* | *Left* | *Right* |
|---|---|---|---|---|
| *Front* | 10<br>-10 | 10<br>-10 | 6<br>-6 | 10<br>-10 |
| *Back* | 8<br>-8 | 7<br>-7 | 5<br>-5 | 7<br>-7 |
| *Left* | 3<br>-3 | 5<br>-5 | 7<br>-7 | 5<br>-5 |
| *Right* | 8<br>-8 | 7<br>-7 | 5<br>-5 | 7<br>-7 |

Interprétation: Q-learning pessimiste : $Q_i(s,a) = \min_o Q_i(s,a,o)$

Problèmes:

- Doit visiter indéfiniment tous les triplet $(etat/a_1/a_2)$ pour converger
- Politique déterministe
- Seulement 2 joueurs

# WoLF-PHC

**Motivation**: Améliorer directement la politique

**Policy Hill Climbing (PHC)**:

- Idée similaire aux *Actor-Critic*
- *Critic* : Q-learning standard

$$Q_i(s, a_i) = Q_i(s, a_i) + \alpha[r_t + \gamma \max_{a_i'} Q_i(s', a_i') - Q_i(s, a_i)]$$

- *Actor* : $\nearrow$ proba de choisir l'action qui maximise Q

$$\pi_i(s, a) = \pi_i(s, a) + \begin{cases} \delta & si \quad a = argmax_{a'} Q(s, a') \\ \frac{-\delta}{|A_i|-1} & sinon \end{cases}$$

## Win or Learn Fast (WoLF)

Principe d'apprentissage selon lequel un agent qui gagne mettra moins vite à jour son modèle ($\delta_{win}$ & $\delta_{lose}$).

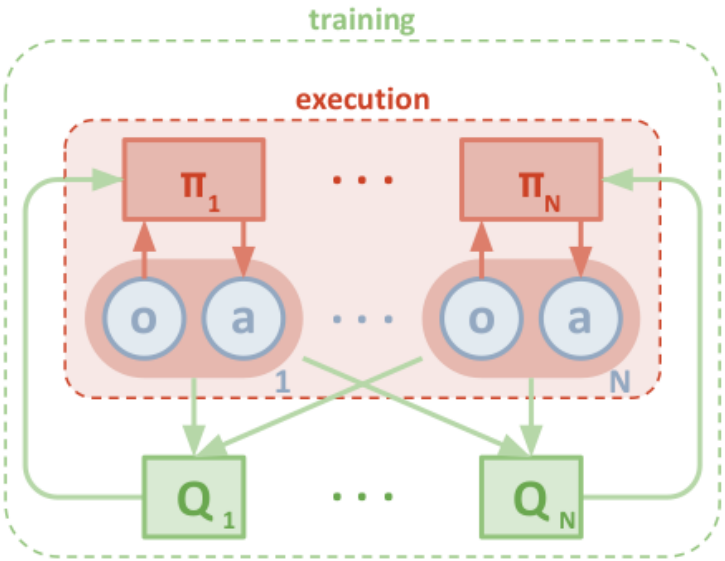
# MADDPG

**Motivation**: Nouvelles méthodes de gradient de politique

**Multi-agent Deep Deterministic Policy Gradient (MADDPG)**:

- Politiques déterministes/Actions continues
- *Critic:* Deep Q-learning
- *Actor:* Descente de gradient

$$\nabla_{\theta_i} J = \frac{1}{S} \sum_j \nabla_{\theta_i} \mu_i(o_i^j) \; \nabla_{a_i} Q_i(s^j, a_1^j, ..., a_i^j, ..., a_N^j)|_{a_i = \mu_i(o_i)}$$



# Présentation de l'API *marl*

- Vision généralisée de l'apprentissage multi-agent
- Système d'agents **"mixtes"** :
  - Agents entraînables, Agents pré-entraînés, Bots, Humains, etc
- Algorithmes disponibles:

**Single-agent algorithms**

| Q-learning | DQN | Actor-Critic | DDPG | TD3 |
|---|---|---|---|---|
| ✔ | ✔ | ✔ | ✔ | ✘ |

**Multi-agent algorithms**

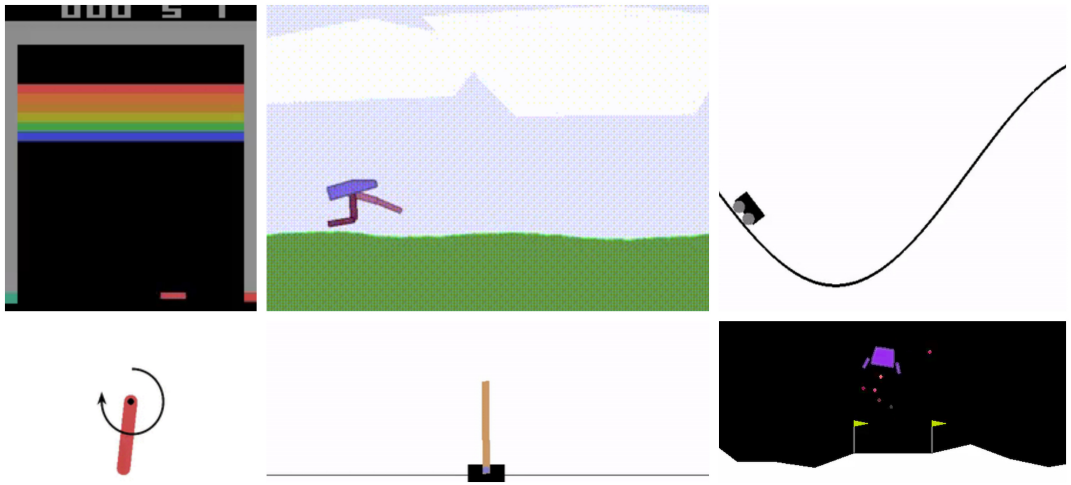
| minimaxQ | PHC | JAL | MAAC | MADDPG |
|---|---|---|---|---|
| ✔ | ✔ | ✘ | ✔ | ✔ |

- Documentation sphinx: `https://blavad.github.io/marl`

- Formalisme de jeux stochastiques
  - 1 récompense / agent
  - pas de communication explicite
- Compatiblilité : environnements *Gym*

# L'environnement : la clé de l'apprentissage

- Environnements *Gym*
  - 2 attributs : *observation_space*, *action_space*
  - 3 méthodes : *reset()*, *step(action)*, *render()*



- Jeux stochastiques $\Leftrightarrow$ à priori non-coopératif
  - Formalisme coopératif : $r_i = r_j \ \forall i, j$

- Jeux stochastiques $\Leftrightarrow$ pas de communication explicite
  - Communication implicite de $i$ vers $j$ : $o_j^{(t+1)} = concat(c_j^{(t+1)}, a_i^{(t)})$

# L'environnement *Soccer*

Deux types d'environnements:

1) Environnement discret
2) Environnement continue

# L'environnement *Soccer*

Deux types d'environnements:

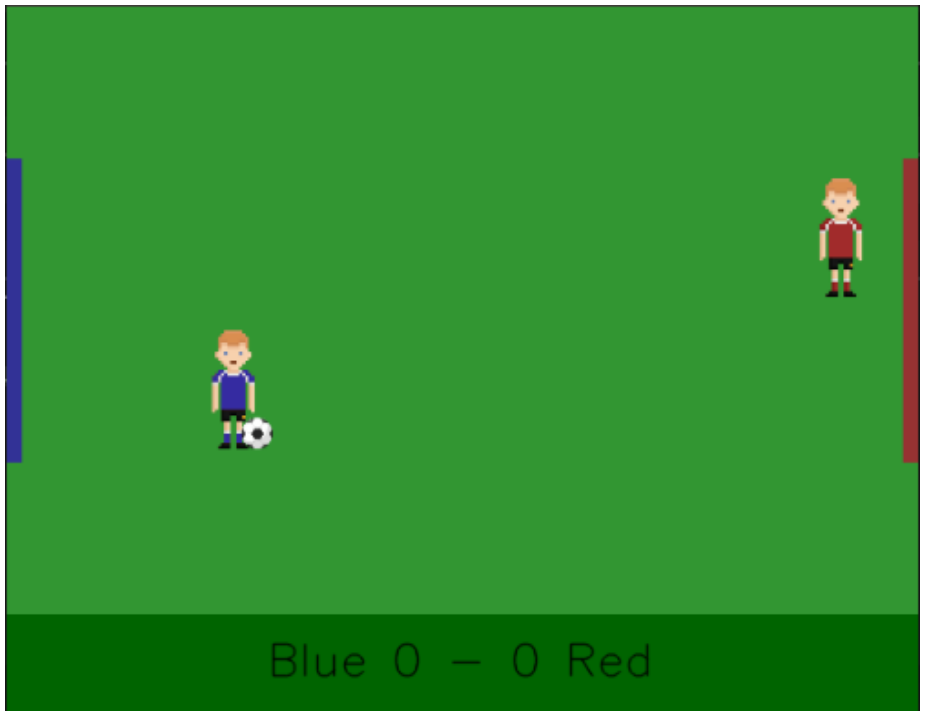
1) Environnement discret

→ Grille $h \times w$ (ici $4 \times 5$)

→ Action : $a \in \{none, front, back, left, right\}$

→ Deux types d'observations

- Etat exact : $s \in [0, ..., n \times (hw)^n]$
- Map : $s \in \{0,1\}^h \times \{0,1\}^w \times \{0,1\}^3$



# L'environnement *Soccer*

Deux types d'environnements:

2) Environnement continue

→ Action : $a \in \{none, front, back, left, right\}$

→ Observation : $o \in [-1, 1]^{5+2\times n}$

# Expérimentations → *Q-learning, minimax-Q et PHC*

## Soccer 2x2

Minimax-Q vs Random

- Minimax-Q : 93.6 %
- Random : 9.4 %
- Match nul : $\leq 2\%$

Q-learning vs Random

- Q-learning : 95.9 %
- Random : 4.1 %
- Match nul : $\leq 1\%$

PHC vs Random

- PHC : 82.3 %
- Random : 17.7 %
- Match nul : $\leq 10\%$

Minimax-Q vs Q-learning

- Minimax-Q : 100.0 %
- Q-learning : 0.0 %
- Match nul : $\geq 70\%$

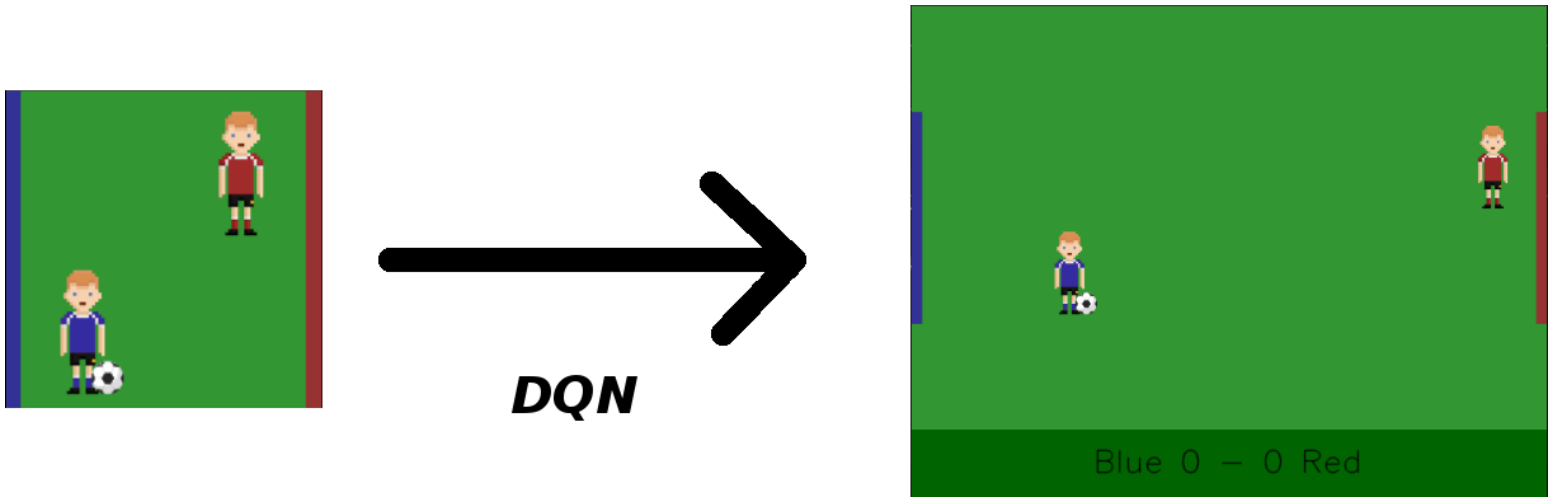
MinimaxQ vs PHC

- Minimax-Q : 58.2 %
- PHC : 41.8 %
- Match nul : 1 %

Problème: Etat de l'environnement devient vite trop grand.

Solution:

- DQN sur terrain 5x5
- CNN pour estimer la valeur $Q$



# Conclusion

- Compétences développées:
  - Théorie sur l'apprentissage multi-agents
  - Programmation python et pytorch
  - Compréhension de méthodes de RL récentes (ex: DDPG)
  - Compréhension des problématiques liées au MARL
  - Développement d'un environnement Gym