# IFT 6756 - Lecture 21
# (Learning in Multi-Agent Systems)

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

**Scribes** **Instructor:** Gauthier Gidel
**Winter 2021:** [Harsh Kumar Roshan, Rupali Bhati, Sree Rama Sanjeev Pratti]

## 1 Summary

This lecture discusses in detail about learning in Multi-Agent systems, is based on the paper **"Open-ended learning in symmetric zero-sum games."** *International Conference on Machine Learning*. PMLR, 2019 [3]

The functional form for Anti-Symmetric (Zero Sum) Game is introduced, Self-play dynamics are discussed together with Elo Rating.

## 2 Motivation

In single-player games, the performance is hand-crafted and defined by the user whereas in multi-player games, the performance is dependant on the opponent(s). Therefore, achieving super-human performance in multi-player games is very challenging and requires a deep understanding of the game.

## 3 Anti-Symmetric (Zero-Sum) Game Functional Form

**Functional-form games (FFGs)**. We derive the Functional form as stated by [3], by computing the probability of one beating the other agent in a game, for any given pair of agents. Few examples are Go, Chess, or StarCraft. The definition stated in [3] further elaborates.

**Definition 1.** *Let $W$ be a set of agents parameterized by, say, the weights of a neural net. A symmetric zero-sum functional-form game (FFG) is an antisymmetric function, that evaluates pairs of agents.*

$$\varphi : W \times W \to \mathbb{R} \tag{1}$$

*The higher $\varphi(v, w)$, the better for agent $v$. We refer to $\varphi > 0$, $\varphi < 0$, and $\varphi = 0$ as wins, losses and ties for $v$.*

This form can also be known as symmetric zero-sum functional-form game (FFG) with an anti-symmetric function,

$$\varphi(v, w) = -\varphi(w, v)$$

In these lecture notes, we use the term anti-symmetric to be consistent with the video lectures instruction. This formulation can also be intuitively understood, if the agents switch their roles, the results are also switched. Few examples of such games include Chess, Poker, Go etc., but agents start is randomized. This formulation can also be generalized for non-zero sum games.

The below important points are to be noted given in [3].

1. The strategies in a FFG are *parameterized agents*, but can be anything that plays the game: (e.g., chess engine, human players)
2. The parameterization of the agents is given by $\phi$, which also shows the composite nature of the game by including both the agent's architecture and the environment.

Suppose the probability of $v$ beating $w$, denoted $P(v \succ w)$ can be computed or estimated. Win/loss probabilities can be rendered into anti-symmetric form via,

$$\varphi(v, w) := P(v \succ w) - \frac{1}{2} \tag{2}$$

$$\varphi(v, w) := \log \frac{P(v \succ w)}{P(v \prec w)} \tag{3}$$

### 3.1 Interpretation of Pay-Off

From the above formulation, the pay-off can be interpreted in the context of two-player game as follows:

$$\varphi(v, w) > 0, \;\; v \;\; beats \;\; w$$

$$\varphi(v, w) < 0, \;\; w \;\; beats \;\; v$$

$$\varphi(v, w) = 0, \;\; It \;\; is \;\; a \;\; \textbf{Tie}$$

## 4 Transitive Games

Due to the continuous nature of the FFGs, different methods have been devised.

*Agents to Objectives*: The equation given below from [1] in [3] illustrates, the conversion of Agents to Objectives with a **curry** operator.

$$\begin{array}{ccc} \big[\phi := W \;\; \times \;\; W \to \mathbb{R}\big] & \xrightarrow{\text{curry}} & \big[W \to [W \to \mathbb{R}]\,\big] \\ \phi(v, w) & & w \mapsto \phi_w(\bullet) := \phi(\bullet, w) \end{array} \tag{4}$$

Referring to the equation above, given the agent $v$ and objective $\phi_w(\bullet)$, new agent $v^1$ can be defined as below:

$$v^1 \;\; := Oracle(v_t, \varphi_{v_t}(\bullet)) \;\;\; \text{when,} \;\; \varphi_w(\bullet) > \varphi_w(v) + e \tag{5}$$

The oracle most commonly uses the gradients or any other reinforcement learning or evolutionary algorithms.

*Anti-Symmetric Evaluation Matrix*: Given a population $\beta$ of *n* agents, the (*n* x *n*) anti-symmetric evaluation matrix is given as:

$$A_\beta := \big\{\phi(\mathbf{w}_i, \mathbf{w}_j) \colon (\mathbf{w}_i, \mathbf{w}_j) \in \beta \times \beta\big\} =: \phi(\beta \otimes \beta) \tag{6}$$

**Nash Equilibrium** on the zero-sum matrix game specified by $A_\beta$. The **Nash Equilibrium** is not necessarily unique. Finally, we use the following **game decomposition** as mentioned in 2. Suppose W is a compact set equipped with a probability measure. The set of integrable antisymmetric functions on W then forms a vector space.

**Theorem 2** (Game decomposition)**.** *Every FFG decomposes into a sum of a transitive and cyclic game*

$$FFG \;\; = \;\; \textit{transitive game} \;\; \otimes \;\; \textit{cyclic game}$$

*with respect to a suitably defined inner product*

***Transitive Games***: A game is transitive if there is a 'rating function' f such that performance on the game is the difference in ratings:

$$\phi(v, w) = f(v) - f(w). \tag{7}$$

In other words, if $\phi$ admits a 'subtractive factorization'.

**Optimization.** Therefore, [3] summarized that solving a transitive game is equivalent to finding,

$$\mathbf{v}^{\star} \ := \ \arg\max_{v \in W} \phi_w(v) = \ \arg\max_{v \in W} f(v) \tag{8}$$

From the above equation, it can be understood that the choice is **w** does not impact the solution. Moreover, the below figure illustrates a simplest algorithm given in [3], to train against a fixed opponent.

**Algorithm 1** Optimization (against a fixed opponent)

```
input: opponent w; agent $v_1$
fix objective: $\phi_w(\bullet)$
for t = 1, . . . , T do
    $v_{t+1} \leftarrow Oracle(v_t, \phi_w(\bullet))$
end for
output: $v_{T+1}$
```

***Monotonic Games***; These games generalize the ***transitive games***. Thus, a FFG is said to be monotonic if there is a monotonic function $\sigma$ such that

$$\phi(\mathbf{v}, \mathbf{w}) = \sigma(f(\mathbf{v}) - f(\mathbf{w})) \tag{9}$$

## 4.1 Elo Rating

The above equation 9 gives way to Elo Rating as stated in [4], which models the probability of one agent beating other as,

$$P(\mathrm{v} \succ \mathrm{w}) = \sigma(f(\mathbf{v}) - f(\mathbf{w})) \tag{10}$$

for,

$$\sigma(x) \ = \ \frac{1}{1 + e^{-\alpha x}} \tag{11}$$

for some $\alpha > 0$, where $f$ assigns Elo ratings to agents. Games such as Chess, Go mostly use this model.
[1] stated that optimizing against a fixed opponent fares badly in monotonic games.

In the context of Elo's model, training against a weaker opponent yields no learning signal because the gradient vanishes,

$$\nabla_{\mathbf{v}} \phi(\mathbf{v}_t, \mathbf{w}) \approx 0 \tag{12}$$

once the sigmoid saturates in the case of

$$f(\mathbf{v}_t) \gg f(\mathbf{w}) \tag{13}$$

## 4.2 Open-ended Learning

General Framework to answer the question: "Who plays against who?"
The way to abstract that is to assume that we have an access to oracle, that when given an agent, an opponent and a payoff. We get an updated agent.

updated agent $\leftarrow$ oracle(agent, opp, payoff)
such that, $\phi$(updated agent, opponent) $>$ $\phi$(agent, opponent)

Now the question is who should be the opponent in order to have an updated agent stronger?
Which can be give as the below equation of Self-play

$$u_{t+1} \leftarrow oracle(u_t, u_t, \phi) \tag{14}$$

Example for oracle,

1. Gradient Descent method:

$$\text{oracle}(u_t, v_t, \phi) = u_t + \eta \nabla_u \phi(u_t, v_t) \tag{15}$$

2. RL Algorithms(Gradient based or not). For instance Q-learning:

$$\text{oracle}(Q_u, Q_v, \phi)(s, a) = Q_u(s, a) + \eta(r_v + \gamma \max_a Q_u(s^+, a) - Q(s, a)) \tag{16}$$

   where $Q_v$ and $r_v$ is reward against $v$.

3. Evolutionary Algorithms

**Conclusion.**

- General framework to understand general algorithm such as self-play or Fictitious self play.

## 4.3 Self-Play

Self play is an algorithm where a player play against a copy of yourself. The idea behind Self play is having a Well collaborated opponent. Elo-rating can help in understanding why Self-play is important.

This algorithm has been proven successful in Chess, Go and many other applications.

An issue with Self-play is that it assumes that payoff is transitive.

$$\phi(v_{t+1}, v_t) > 0, ..., \phi(v_1, v_0) > 0 \Rightarrow \phi(v_{t+1}, v_i), i \in [t] \tag{17}$$

An idea behind improvement of $v_t$ implies global improvement.

## 4.4 The Bilinear Game

The various versions of Bilinear game are:

- 2-D version of 1-D bilnear example, **Simple Payoff**:

$$\phi(u, w) = u_1 w_2 - w_1 u_2 \tag{18}$$

- Self-Play:

$$u_{t+1} = u_t + \eta \nabla_u \phi(u_t, \widetilde{u_t}) \tag{19}$$

which is in practice and can be written as:

$$\theta_{t+1} = \theta_t + \eta \frac{d\phi(u_{\theta_t}, u_{\widetilde{\theta_t}})}{dt} \tag{20}$$

**Proposition.** The dynamic of self play diverges.

The vector field depends on your opponent. At each timestamp the vector field is orthogonal to the direction to the optimum.

### 4.5 Playing Against a Group of Agents

Consider a population (group) of agents $B = (u_i)$. We can consider the payoff of this group of agents as $A_B$. The matrix $A_B$ is anti-symmetric because the payoff is anti-symmetric.

$$[A_B]_{ij} = \varphi(u_i, u_j) \tag{21}$$

As shown in Fig.1, the payoff can indicate what kind of game is being played. In the "Almost Transitive" case, there is a green agent at the bottom left corner of the grid which can beat almost every other agent. Similarly, at the top right corner, there exists a red player which is beaten by almost every other player. Hence, making this an almost transitive game. The second row of Fig.1 indicates the ranking among the agents.

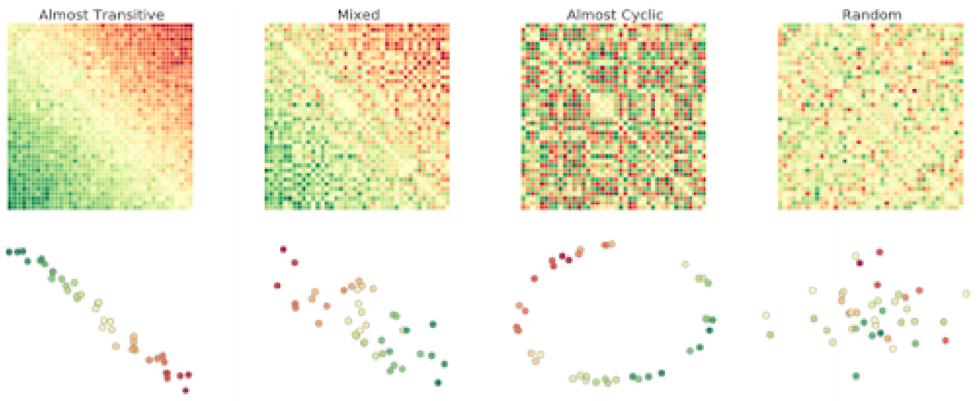

Figure 1: Different types of Games based on the payoff matrix [2]

### 4.6 Nash of an Empirical Game

Suppose there exists a group of players that are the best players, we wish to find the Nash equilibrium for which these players exist.

**Proposition 3.**
$$Nash = \left\{P : P^T \cdot A \geq 0, P \geq 0\right\} \tag{22}$$

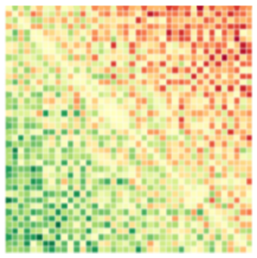

Figure 2: Pay-Off Matrix for Mixture of Agents[2] ; Sample $V_i$ with Probability $P_i$

### 4.7 Matrix of the empirical game

This matrix can be used for several purposes:

1. It can help in evaluating agent or a group of agents.
2. It can be used to evaluate the diversity of a group of agents.

3. Also it can be used to setup efficient training.

**Evaluating the performance of a population.** The below definition and proposition from [3] can help us in evaluating the performance of a population.

**Definition 4.** *Given populations $\mathfrak{B}$ and $\mathfrak{Q}$, let $(\mathbf{p}, \mathbf{q})$ be a Nash Equilibrium of zero-sum game on $\mathbf{A}_{\mathfrak{B},\mathfrak{Q}} := \phi(\mathbf{v}, \mathbf{w})_{\mathbf{v}\in\mathfrak{B},\mathbf{w}\in\mathfrak{Q}}$. The **relative population performance** is*

$$v(\mathfrak{B}, \mathfrak{Q}) := \mathbf{p^T} \cdot \mathbf{A}_{\mathfrak{B},\mathfrak{Q}} \cdot \mathbf{q} = \sum_{i,j=1}^{n_1,n_2} A_{i,j} \cdot p_i q_j \tag{23}$$

**Proposition 5.** *(i) Performance $v$ is independent of the choice of Nash equilibrium.*

*(ii) if $\phi$ is monotonic then performance compares the best agents in each population*

$$v(\mathfrak{B}, \mathfrak{Q}) = \max_{\mathbf{v}\in\mathfrak{B}} f(\mathbf{v}) - \max_{\mathbf{w}\in\mathfrak{Q}} f(\mathbf{w}) \tag{24}$$

*(iii) If hull($\mathfrak{B}$) $\subset$ hull($\mathfrak{Q}$) then $v(\mathfrak{B}, \mathfrak{Q}) \leq 0$ and $v(\mathfrak{B}, \mathfrak{R}) \leq v(\mathfrak{Q}, \mathfrak{R})$ for **any** population of $\mathfrak{R}$.*

*We can say that, For any population, $v(\mathfrak{B}, \mathfrak{B}) = 0$*

Sanity tests are the first two properties. Growing the polytope spanned by a population increases its success against every other population, according to property (iii).

**Evaluate Density of the population** Below definition as in [3] can be used to evaluate the density of population.

**Definition 6.** *Denote the rectifier by $\lfloor x \rfloor_+ := x$ if $x \geq 0$ and $\lfloor x \rfloor_+ := 0$ otherwise. Given population $\mathfrak{B}$, let $\mathbf{p}$ be a Nash equilibrium on $\mathbf{A}_{\mathfrak{B}}$. The **effective density** of the population is:*

$$d(\mathfrak{B}) := \mathbf{p_T} \cdot \lfloor \mathbf{A}_{\mathfrak{B}} \rfloor_+ \cdot \mathbf{p} = \sum_{i,j=1}^{n} \lfloor \phi(\mathbf{w}_i, \mathbf{w}_j) \rfloor_+ \cdot p_i p_j \tag{25}$$

The best agents (those with help in the highest entropy Nash) exploit each other in a diversity. If a dominant agent exists, diversity is zero.

**How to train agents Efficiently?** We can use this matrix to find who to train the agent against. The options are:

- To train against the Nash.
- To train against the best Response.

**Idea: Algorithm** This algorithm creates a series of fruitful local priorities that, once solved, add up to transitive population-level success iteratively. This algorithm, unlike self-play, produces communities rather than single agents.

**Algorithm 2** Response to Nash ($PSRO_N$)

```
input: population $\mathfrak{B}_1$ of agents
for $t$=1,...,$T$ do
    $p_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{B}_t}$
    $v_{t+1} \leftarrow$ oracle$\big(v_t, \sum_{w_i\in\mathfrak{B}_t} \mathbf{p}_t[i] \cdot \phi_{w_i}(\cdot)\big)$
    $\mathfrak{B}_{t+1} \leftarrow \mathfrak{B}_t \cup \{v_{t+1}\}$
end for
output: $\mathfrak{B}_{T+1}$
```

$PSRO_N$ (policy space reaction to the Nash) produces new agents that are estimated best responses to the Nash mixture iteratively. $PSRO_N$ degenerates into self-play if the game is transitive.
**Problem:** An issue with this algorithm is that sometimes it calculates zero gradient(e.g., Bilinear example).

## 4.8 Fictitious Self-Play

In this algorithm the player plays against an opponent which is better than the player. Given $v_i$ as the group of agents, we can represent it as,

$$u_{t+1} \leftarrow \text{oracle}(u_t, \text{best opp}, \phi) \tag{26}$$
$$\text{best opp} := \arg\min_{v_i} \phi(u, v_i) \tag{27}$$

This algorithm is used in Startcraft II.

## 4.9 Conclusion

- In a Multi-Agent setting, self-play is a very effective way to train agents.
- It doesn't always work (when we need a diversity of agents to play the game)
- We can use the empirical payoff when dealing with a group of agents to:
  - Evaluating agents
  - Training agents
  - Evaluation of the group (performance and diversity)

## References

[1] H. Aziz. Multiagent systems: Algorithmic, game-theoretic, and logical foundations by y. shoham and k. leyton-brown cambridge university press, 2008. *SIGACT News*, 41(1):34–37, Mar. 2010. ISSN 0163-5700. doi: 10.1145/1753171.1753181. URL https://doi.org/10.1145/1753171.1753181.

[2] D. Balduzzi, K. Tuyls, J. Perolat, and T. Graepel. Re-evaluating evaluation, 2018.

[3] D. Balduzzi, M. Garnelo, Y. Bachrach, W. M. Czarnecki, J. Pérolat, M. Jaderberg, and T. Graepel. Open-ended learning in symmetric zero-sum games. *CoRR*, abs/1901.08106, 2019. URL http://arxiv.org/abs/1901.08106.

[4] A. E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978. ISBN 0668047216 9780668047210. URL http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216.