Designing Learning Algorithms over the Sequence Form of an Extensive-Form Game

(Extended Abstract)

Edoardo Manino
University of Southampton
em4e15@soton.ac.uk

Nicola Gatti
Politecnico di Milano
nicola.gatti@polimi.it

Marcello Restelli
Politecnico di Milano
marcello.restelli@polimi.it

ABSTRACT

We focus on multi-agent learning over extensive-form games. When designing algorithms for extensive-form games, it is common the resort to tabular representations (i.e., *normal form*, *agent form*, and *sequence form*). Each representation provides some advantages and suffers from some drawbacks and it is not known which representation, if any, is the best one in multi-agent learning. In particular, a wide literature studies algorithms for the normal form, but this representation is prohibitive in practice since it is exponentially large in the size of the game tree. In this paper, we show that some learning algorithms defined over the normal form can be re-defined over the sequence form so that the dynamics of the two algorithms are *realization equivalent* (i.e., they induce the same probability distribution over the outcomes). This allows an exponential compression of the representation and therefore makes such algorithms employable in practice.

Keywords

Extensive-form games; sequence form; multi-agent learning.

1. INTRODUCTION

Multi-agent learning is a long-standing challenging problem in the multi-agent community. A classic model is to have the agents learn the strategy of a one-shot game through repeated interaction. Hence, the goal is designing algorithms that are able to converge to an equilibrium in self-play, and exploit at best the opponent's strategy when paired against other algorithms. A large portion of the literature mainly investigates games where the players act simultaneously (aka normal-form games), whereas when the interaction unfolds over several consecutive actions (extensive-form games) many issues are left open. As we argue below, in order to work on this more complex case it is necessary to cast the game to one of several possible alternative representations. Nowadays it is not known which one is the best.

State of the art. The first option is to transform the extensive-form game into its normal-form equivalent, thus allowing the access to a large collection of algorithms including Cross' Learning [3], GIGA [19], Exp3 [1] and $\mathcal{Q}$-learning [17]. For many of these, we have a full characterisation of their learning dynamics, which is tipically modeled as a *replicator dynamics* with a specific mutation term [13, 7, 18] or as a gradient ascent algorithm [8]. The main drawback is that the normal form is exponentially large in the size of the game tree, thus making the learning process exponentially long as the algorithm has to sort through a large number of alternative plans.

The second option is to use the agent form, where a different fictitious agent plays at each information set. An example that is closely related to this form is the *counterfactual regret* (CFR) minimization algorithm [20], which has been successfully applied to very large games (e.g. Poker) [11]. CFR is known to converge to a Nash equilibrium in two-player zero-sum games [20] and has been evaluated on other categories of games [9, 12]. In the case of general-sum games, it is only known that strictly dominated actions will be played with probability zero [6], but no result characterizing its dynamics is known. More in general, the main drawback of the agent form is that learning is performed independently at each information set. As a result, even though the strategy space is exponentially smaller with respect to the normal-form, it is possible to have more chaotic learning dynamics, and long delays in the propagation of the optimal behaviour along the tree.

Finally, there is a third option that involves the use of the sequence-form instead. This less-common representation does not require fictitious agents and is linear in the size of the game tree. As an example, Leduc Hold'em, a small variant of Poker, has 337 sequences per player compared to about 10^{14} different actions in the normal-form. On the other side, its strategy space is not a simplex anymore, but a convex set defined by a set of linear constraints. The first applications of this approach in multi-agent learning are described in [5, 10] where a sequence-form replicator dynamics and a $\mathcal{Q}$-learning algorithm are introduced respectively. However, as remarked in [2], the use of the sequence form in multi-agent learning is largely unexplored.

Original contributions. In this paper, we provide new results on multi-agent learning over the sequence form. More precisely, we propose the sequence-form as a bridge to regain access to the collection of normal-form algorithms for extensive-form games. To this end, we design the sequence-form equivalent of three algorithms, each based on a different principle: a replicator dynamics algorithm (Cross' Learning), a gradient ascent one (GIGA) and a bandit one (Exp3). In doing so, we show how to solve a recurring technical issue, i.e. the problem of efficiently counting the number of normal-form plans that include a specific sequence.

2. PRELIMINARIES

We focus here on the description of the normal-form and the sequence-form. For a definition of extensive-form game see [4].

(Reduced) Normal form This is a tabular representation [15] in which each plan $p \in P_i$ specifies one action a per information set. We denote by σ_i the *mixed strategy* of player i, which specifies the

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

Algorithm 1 PlanCount(i, Γ, x)

```
1:  if $x \in Q_i$ then
2:      if $x$ is a terminal sequence then
3:          return 1
4:      else
5:          $\mathsf{temp} = 1$
6:          for $h \in H_i$ s.t. $\exists a \in \rho(h) : xa \in Q_i$ do
7:              $\mathsf{temp} \leftarrow \mathsf{temp} \cdot \mathsf{PlanCount}(i, \Gamma, h)$
8:  if $x \in H_i$ then
9:      $\mathsf{temp} = 0$
10:     for $a \in \rho(x)$ do
11:         $\mathsf{temp} \leftarrow \mathsf{temp} + \mathsf{PlanCount}(i, \Gamma, q(x)a)$
12: return temp
```

probability $\sigma_i(p)$ associated with each plan p. The *reduced* normal-form is obtained by deleting replicated strategies [14], therefore decreasing the total number of plans $|P_i|$. Although the reduced normal form can be much smaller than normal form, it is still exponential in the size of the game tree.

Sequence form This representation is constituted by a sparse payoff matrix and a set of constraints [16]. A sequence $q \in Q_i$ is a set of consecutive actions $a \in A_i$ by player i. We denote by $q_\emptyset$ the initial sequence of every player, by qa the *extended* sequence obtained by appending action a to sequence q, and by $q(h)$ the sequence leading to information set $h \in H_i$. Finally we denote by r_i the sequence-form strategy of player i. Well-defined strategies are such that, for every information set $h \in H_i$, we have the constraint $r_i(q) = \sum_a r_i(qa)$ where $q = q(h)$ and $a \in h$. The player i's utility is represented as a sparse multi-dimensional array, denoted, with an abuse of notation, by U_i, specifying the value associated with every combination of sequences of all the players.

Realization equivalence Two strategies σ_i and r_i are realization equivalent if they induce the same probability distribution over the outcomes for every strategy of the opponents. Given a game and a player i, a sequence-form strategy r_i and a normal-form strategy σ_i are realization equivalent if and only if, for every $qa \in Q_i$, $r_i(qa) = \sum_{p \in P_i : a \in p} \sigma_i(p)$ holds.

3. REDEFINING ALGORITHMS OVER THE SEQUENCE FORM

In this section we present a summary of our results. First we explain how to produce a *uniform strategy*, that is a strategy realization equivalent to $\sigma_i(p) = 1/|P_i|$. Then we move to the details of our multi-agent learning algorithms.

Uniform Strategy By definition of realization equivalence we have:

$$r_i^{uni}(qa) = \sum_{p \in P_i : a \in p} \sigma_i(p) = \frac{1}{|P_i|} \sum_{p \in P_i : a \in p} 1 \qquad (1)$$

However, counting the number of plans p that include the sequence qa has exponential complexity in the size of the game tree. We propose instead a procedure that can compute the result in polynomial time (see Algorithm 1): for each sequence qa, call PlanCount$(i, \text{Restrict}(\Gamma, qa), x)$, where Restrict$(\Gamma, qa)$ returns a modified game Γ' in which player i always plays qa if possible.

SF-Cross' Learning We recall that the normal-form Cross' Learning [3] is initialised to the uniform strategy and then updated according to:

$$\sigma_i^{t+1} = (1 - R_i^t)\sigma_i^t + R_i^t e_{\hat{p}} \qquad (2)$$

where R_i^t is the reward obtained by playing action $\hat{p}$. The sequence-form version of this algorithm is initialized to the uniform strategy $r_i^0 = r_i^{uni}$ as well. Then, at every time-step t we extract from r_i^t a pure realization-plan $\hat{r}_i^t$ (see [10]), play according to it, and receive the reward R_i^t. The strategy is then updated with Cross' original formula, though over the sequence-form:

$$r_i^{t+1}(q) = (1 - R_i^t) r_i^t(q) + R_i^t \hat{r}_i^t \qquad (3)$$

In the normal form, the average learning dynamics of Cross' Learning converge to the continuous-time replicator dynamics. We can show the same thing over the sequence form, as the expected change in strategy $r_i^{t+1} - r_i^t$ when $t \to 0$ is:

$$\mathbb{E}[\Delta r_i^{t+1}(q) | r_{-i}^t] = r_i^t(q)(g_q(r_i^t) - r_i^t) U_i r_{-i}^t \qquad (4)$$

which is the aforementioned equation of the continuous-time replicator dynamics in sequence form [5].

SF-GIGA We recall that the strategy update of the GIGA algorithm in normal form is as follows [19]:

$$\sigma_i^{t+1} = \mathcal{P}\left(\sigma_i^t + \eta U_i e_{\hat{p}_{-i}}^t\right) \qquad (5)$$

where the gradient $U_i e_{\hat{p}_{-i}}^t$ depends on the pure strategy $\hat{p}_{-i}$ of the opponent(s), and $\mathcal{P}$ projects back the result in the strategy space. It must be noted that $\mathcal{P}$ can be written as a linear operator as long as σ_i^t is in the inner of the strategy space. Under this circumstances, we can write a realization equivalent form of GIGA:

$$r_i^{t+1}(qa) = r_i^t(qa) + \eta(z_{qa} U_i \hat{r}_{-i}^t - \bar{w}(qa)) \qquad (6)$$

where the vector z_{qa} is the sum of all the possible pure strategies $\hat{r}_i$ that contain qa, and $\bar{w}(qa)$ is the expected utility of player i weighted by the ratio of plans insisting on qa. Both are again a matter of counting that can be solved in polynomial time as follows:

$$z_{qa}(q'a') = \text{PlanCount}(i, \text{Restrict}(\text{Restrict}(\Gamma, q'a'), qa), q_\emptyset)$$

$$\bar{w}(qa) = \left(l_i U_i r_{-i}^t\right) \frac{\text{PlanCount}(i, \text{Restrict}(\Gamma, qa), q_\emptyset)}{\text{PlanCount}(i, \Gamma, q_\emptyset)}$$

where $l_i(q) = \text{PlanCount}(i, \text{Restrict}(\Gamma, q), q_\emptyset)$.

SF-Exp3 We recall that the normal-form Exp3 algorithm [1] defines player's i strategy at time t as follows:

$$\sigma_i^t(p) = (1 - \gamma) \frac{\alpha^{s^t(p)}}{\sum_{p' \in P_i} \alpha^{s^t(p')}} + \frac{\gamma}{|P_i|} \qquad (7)$$

where $\alpha > 1$, $\gamma \in (0, 1)$ regulates the amount of uniform exploration, and $s^t(p)$ is the sum of all the rewards obtained by playing the plan p in the past. Given the definition of realization equivalence, the sequence-form version of Exp3 follows:

$$r_i(qa) = (1 - \gamma) \frac{\sum_{p \in P_i : a \in p} \alpha^{s^t(p)}}{\sum_{p' \in P_i} \alpha^{s^t(p')}} + \gamma r_i^{uni} \qquad (8)$$

Notice that we only need the parameters s^t for the plans p effectively played during the game, hence achieving an $O(T)$ memory requirement. Furthermore, we can incrementally update the term $\sum_{a \in p} \alpha^{s^t(p)}$ after every round t, thus keeping the time complexity linear in the size of the game tree.

4. FUTURE WORK

In future, we will study the online version of the GIGA algorithm defined over the sequence form, and we will compare its performance w.r.t. the other online algorithms, including the online version of the CFR. Furthermore, we will develop the sequence-form version of other algorithms, among which the ϵ-greedy Q-learning algorithm and the WoLF-GIGA.

REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-arm bandit problem. In *FOCS*, pages 322–331, 1995.

[2] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers. Evolutionary dynamics of multi–agent learning: A survey. *J ARTIF INTELL RES*, 53:659–697, 2015.

[3] J. G. Cross. A Stochastic Learning Model of Economic Behavior. *The Quarterly Journal of Economics*, 87(2):239–266, 1973.

[4] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991.

[5] N. Gatti, F. Panozzo, and M. Restelli. Efficient evolutionary dynamics with extensive–form games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 335–341, Bellevue, Washington, USA, July 14–18 2013.

[6] R. Gibson. Regret minimization in non-zero-sum games with applications to building champion multiplayer computer poker agents. *arXiv preprint arXiv:1305.0034*, 2013.

[7] E. R. Gomes and R. Kowalczyk. Dynamic analysis of multiagent q–learning with ϵ–greedy exploration. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 369–376. ACM, 2009.

[8] M. Kaisers, D. Bloembergen, and K. Tuyls. A common gradient in multi–agent reinforcement learning. In *AAMAS*, pages 1393–1394. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[9] M. Lanctot, R. Gibson, N. Burch, M. Zinkevich, and M. Bowling. No-regret learning in extensive-form games with imperfect recall. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML 2012)*, 2012.

[10] F. Panozzo, N. Gatti, and M. Restelli. Evolutionary dynamics of Q–learning over the sequence form. In *AAAI*, pages 2034–2040, 2014.

[11] M. Ponsen, S. de Jong, and M. Lanctot. Computing approximate nash equilibria and robust best-responses using sampling. *Journal of Artificial Intelligence Research*, 42(1):575–605, 2011.

[12] N. A. Risk and D. Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 159–166. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[13] K. Tuyls, P. J. Hoen, and B. Vanschoenwinkel. An evolutionary dynamical analysis of multi–agent learning in iterated games. *Autonomous Agents and Multi-Agent Systems*, 12(1):115–153, 2006.

[14] D. Vermeulen and M. Jansen. The reduced form of a game. *EUR J OPER RES*, 106(1):204–211, 1998.

[15] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.

[16] B. von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

[17] C. J. Watkins and P. Dayan. Q–learning. *Machine learning*, 8(3–4):279–292, 1992.

[18] M. Wunder, M. L. Littman, and M. Babes. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1167–1174, 2010.

[19] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

[20] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, pages 1729–1736, 2007.