Algorithmic Game Theory — Autumn 2021, Week 13

Very Hard Games

ETH Zürich — Paolo Penna

These notes are quite **informal**, and mainly contain references and few details about the topics discussed in the last (bonus) lecture:

- Generative Adversarial Networks (GANs)
- Games in Statistical Physics and Biology
- Q-learning in Reinforcement Learning

These new topics will not be part of the exam, though we use them to refresh our knowledge about topics seen in the course (and maybe to see the limitations of these results, and what is beyond this course).

1 Generative Adversarial Networks (GANs)

Sources and more details:

- Blog about GANs
- GANs animation/visualization webpage
- YouTube video
- Original paper (existence of "good" equilibria where training is always "successful")
- Older paper (key ingredient about "almost pure" good equilibria – Thm 2)

The main idea og GANs is to simultaneously training a Generator (GEN) and a Discriminator (DIS) by letting them "compete against each other" very much like a zero-sum game. Each player is a neural network and the strategy is the choice of the internal parameters/weights. The generator generates some "image" $x \sim G$ and the discriminator should tell whether it is fake ($D(x) = 0$) or real ($D(x) = 1$). The discriminator gets in input either a real image ($y \sim R$ where R is the real distribution) or a fake image from the generator:

GANs zero-sum game:

$$U(G, D) = E_{x \sim G}[D(x)] + E_{y \sim R}[1 - D(y)] \quad (1)$$

The generator wants to **maximize** this utility (fool the discriminator as much as possible) and of course the discriminator wants to **minimize** it instead.

Exercise 1. *Consider a simpler version of (1) in which we remove the second term (ignore real distribution). What is the best strategy of the discriminator?*

Exercise 2. *What is the best strategy of the generator for the game in (1), assuming $G = R$ is possible? What is the utility utility of the generator in such strategy? What is the minimum possible utility of the generator?*

Here is what happens (details in the original paper):

1. The strategy of each player (neural network) is a choice of the **weights**;
2. The generator can approximate **any** real distribution (Gaussian distributions are the "building blocks" of all distributions);
3. We can assume the weights (strategies) to be **descrete** from a set $\{0, \delta, 2\delta, \cdots, k\delta\}$ as we anyway only want a good approximation;
4. There is always an (approximate) **PNE** where the generator "wins".

The last point is essential to capture the "successful" training: (i) in the end the generator fixes its weights and uses the same weights to generate all realistic images, (ii) the discriminator essentially cannot distinguish these from the realistic ones, and (iii) no other discriminator can do that.

The discriminator has p parameters (weights) and so has the generator.

Both players have a **huge** set of strategies (though finite) of size at most $N = k^p$.

Every finite zero-sum game has an ϵ-approximate MNE where each player uses at most $O(\log N/\epsilon^2)$ strategies (Thm 2 here).

Back to our GANs application,

1. Any mixed strategy over $r = O(\log N/\epsilon^2)$ pure strategies is a distribution over r (deterministic/pure) generators $G_1, \ldots, G_r$; each of them with p parameters.
2. This distribution can be simulated by a pure "super-generator" with p^2 parameters.

any mixed strategy over $r = O(\log N/\epsilon^2)$ pure strategies is a distribution over r generators $G_1, \ldots, G_r$.

For any real distribution R and any discriminator with at most p parameters, there exists a generator with $O(p^2)$ parameters that "fools" the generator (Thm 4.3 here).

2 Statistical Physics

Sources and more details:

- YouTube video (magnetism and temperature)
- YouTube video (Ising model simulation with Python code)
- Lecture notes (from a course at EPFL by Nicolas Macris)
- Markov chains (basics to analyze the "equilibrium" of these dynamics)
- Original paper (application to strategic games and convergence to "good" equilibria)
- Another paper (about the time to converge to "equilibrium")
- Statistical Learning Theory (ETH course by Joachim Buhmann – many more connections between Statistical Physics and ML, and more)

Think about a player who has limited rationality: When facing different alternatives can only distinguish them if the utility is "significantly" different. When instead two alternatives have roughly the same utility, the player will choose one of them with roughly the same probability.

"Noisy" Best-response (single player setting):

$$p_i^{(\beta)}(a) = \frac{e^{\beta u_i(a)}}{Z^{(\beta)}} \qquad Z^{(\beta)} = \sum_{a' \in A_i} e^{\beta u_i(a')} \tag{2}$$

Exercise 3. *Convince yourself that, for $\beta \to \infty$, the distribution in (2) assigns non-zero probability only to actions with optimal (minimum) cost. What happens if there are multiple BR? What happens from $\beta = 0$ instead?*

In games with several players, we consider the following "noisy" best response dynamics: At each step pick a **random player** and let him/her respond according to (2) where instead of $u_i(a)$ we have $u_i(a, s_{-i}^{(t)})$ and $s^{(t)}$ is the current profile (strategies of each player). This process gives a **Markov chain** whose **stationary distribution** ("equilibrium") can be computed in several games.

For any **potential game** the above dynamics converges to a stationary distribution π of the form

$$\pi(s) = \frac{e^{\beta POT(s)}}{Z_{POT}} \qquad Z_{POT} = \sum_{s'} e^{\beta POT(s')} \tag{3}$$

where POT is the potential function of the game:

$$u_i(s_i, s_{-i}) - u_i(s_i', s_{-i}) = POT(s_i, s_{-i}) - POT(s_i', s_{-i}) \tag{4}$$

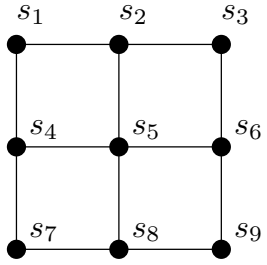
Exercise 4. *Consider the following two coordination games:*

	A	B
A	1, 1	0, 0
B	0, 0	1, 1

	A	B
A	1, 1	0, 0
B	0, 0	2, 2

For $\beta \to \infty$, what is the stationary distribution π in (3) in these games?

The **Ising model** in statistical physics is (essentially) the coordination game on the left played on the simple **grid** network:



3 Biology and Evolution

Sources and more details:

- These lecture notes (from an MIT course)
- Chapter 8 in this book
- These Python examples (part of this course at University of Graz)

In biology we consider a (large) population of individuals, each of them with some **characteristic** (strategy in our terminology). Instead of choosing the strategy, each individual i is matched with a random "mate" and the current strategy of player i is copied proportionally to the **fitness** (utility). We are interested how the population evolves, that is, the **proportion** (probability) of players choosing strategy a over time.

"Replicator" Response:

$$p_i^{(t+1)}(a) = u_i(a) p_i^{(t)}(a) \tag{5}$$

To study the steady states (equilibria) of such dynamics we simply consider the time derivatives and set them to zero.

Replicator dynamics

$$\dot{p}_i^{(t)}(a) = p_i^{(t)}(a)(u_i^{(t)}(a) - \bar{u}_i^{(t)}) \tag{6}$$

where $\bar{u}_i^{(t)} = E_{a\sim p_i^{(t)}}[u_i^{(t)}(a)]$ and $u_i^{(t)}(a) = E_{s_2\sim p_i^{(t)}(a)} u_1(a, s_2)$.

Exercise 5. *Consider the above equation for a single-player setting, and convince yourself that the steady states correspond to the following condition:*

$$p_i^{(t)}(a) > 0 \qquad \Longrightarrow \qquad u_i^{(t)}(a) = \bar{u}_i^{(t)} \tag{7}$$

Try to understand the relation between this condition and MNE.

4 Reinforcement Learning

Sources and more details:

- YouTube video (AlphaGo and others)
- YouTube video (OpenAI multi-agent RL)
- YouTube video (intuition about Q-learning)
- RL Book (with Q-learning in Section 6.5)
- Original paper (analyzing Q-learning in two-players two-strategies games)
- Recent paper (connecting Q-learning to Catastrophe Theory)

The idea of **Q-learning** is to compute the (optimal) value for each **action-state** pair. This can be done via an **iterative procedure** where the **current** estimates $Q^{(t)}$ and the "observed/immediate" reward (utility) give the new estimates $Q^{(t+1)}$:

$$Q^{(t+1)}(a, state) = (1-\alpha)Q^{(t+1)}(a, state) + \alpha[(u_i(a, state) + \gamma \max_b Q^{(t)}(b, state')] \tag{8}$$

where α controls the trade-off between the past estimates and the currently observed utility (this is similar to the learning parameter η is the MWUA algorithm). The other parameter γ is the so-called discount factor, which models that future rewards are "less valuable" than immediate ones (like monetary "inflation"). The right-hand side of (8) assumes that the next action (after going to the new state $state'$) is chosen optimally.

If we consider a "stateless" setting and replace the optimal action choice with a noisy one, we obtain the following "Boltzmann" Q-learning consisting of two parts:

Selection:

$$p^{(t)}(a) = \frac{e^{\beta Q^{(t)}(a)}}{Z(t)} \qquad \text{where } Z(t) = \sum_{a'} e^{\beta Q^{(t)}(a')} \tag{9}$$

Update Q-values:

$$Q^{(t+1)}(a) = (1-\alpha)Q^{(t+1)}(a) + \alpha u_i^{(t)}(a) \tag{10}$$

We look at the time derivatives of the probabilities and using $\frac{de^{yx}}{dx} = ye^{yx}$ we get

$$\dot{p}^{(t)}(a) = p^{(t)}(a) \cdot \beta \left(\dot{Q}^{(t)}(a) - E\dot{Q}^{(t)} \right) \tag{11}$$

where $E\dot{Q}^{(t)} = \sum_{a'} p^{(t)}(a') \cdot \dot{Q}^{(t)}(a')$

Below we show that

$$\dot{Q}^{(t)}(a) = \alpha(u_i^{(t)}(a) - Q^{(t)}(a)) \tag{12}$$

and thus

$$E\dot{Q}^{(t)} = \alpha(Eu_i^{(t)} - EQ^{(t)}) \tag{13}$$

Stationarity condition

$$\dot{Q}^{(t)}(a) = E\dot{Q}^{(t)} \tag{14}$$

which is equivalent to

$$u_i^{(t)}(a) - Eu_i^{(t)} = Q^{(t)}(a) - EQ^{(t)} \tag{15}$$

Since

$$\log p^{(t)}(a) = \beta Q^{(t)}(a) - \log Z(t) \tag{16}$$

we get

$$p^{(t)}(a) - \sum_{a'} p^{(t)}(a) u_i^{(t)}(a) = \frac{1}{\beta} [\log p^{(t)}(a) - \underbrace{\sum_{a'} p^{(t)}(a') \log p^{(t)}(a')}_{-\text{entropy } H(p^{(t)})}] \tag{17}$$

A possible rewriting of this condition is

$$u_i^{(t)}(a) - \frac{1}{\beta} \log p^{(t)}(a) = Eu_i^{(t)} + \frac{1}{\beta} \cdot H(p^{(t)}) \tag{18}$$

The following "noisy" best-response distribution yields a solution:[1]

$$p^{(t)}(a) = \frac{e^{\beta u_i^{(t)}(a)}}{Z^{(\beta)}} \qquad Z^{(\beta)} = \sum_{a' \in A_i} e^{\beta u_i^{(t)}(a')} \tag{19}$$

[1] This is usually called Gibbs distribution.

A Details and Calculations

A.1 Proof of (11)

We first compute the time derivatives of the probabilities in (9) using the following:

$$\frac{\partial}{\partial t}\left(\frac{e^{f(t)}}{g(t)}\right) = e^{f(t)} \cdot \frac{g(t)f'(t) - g'(t)}{g(t)^2} \tag{20}$$

In our case we have

$$\dot{p}^{(t)}(a) \overset{(9)}{=} \frac{\partial}{\partial t}\left(\frac{e^{\beta Q^{(t)}(a)}}{Z(t)}\right) \overset{(20)}{=} e^{\beta Q^{(t)}(a)} \cdot \frac{Z(t) \cdot \beta \dot{Q}^{(t)}(a) - \dot{Z}(t)}{Z(t)^2} \tag{21}$$

$$\overset{(9)}{=} p^{(t)}(a) \cdot \frac{Z(t) \cdot \beta \dot{Q}^{(t)}(a) - \dot{Z}(t)}{Z(t)} = p^{(t)}(a) \cdot \left(\beta \dot{Q}^{(t)}(a) - \frac{\dot{Z}(t)}{Z(t)}\right) \tag{22}$$

and

$$\dot{Z}(t) \overset{(20)}{=} \sum_{a'} e^{\beta Q^{(t)}(a')} \cdot \beta \dot{Q}^{(t)}(a') \Longrightarrow \frac{\dot{Z}(t)}{Z(t)} = \beta \sum_{a'} p^{(t)}(a') \cdot \dot{Q}^{(t)}(a') = \beta E \dot{Q}^{(t)} \tag{23}$$

Thus (11) follows directly by combining the last two equalities.

A.2 Deriving (12)

Start from

$$Q^{(t+1)}(a) - Q^{(t)}(a) = \alpha[u_i^{(t)}(a) - Q^{(t)}(a)] \tag{24}$$

and replace

$$t + 1 \to t + \delta t \qquad \text{and} \qquad \alpha \to \alpha \delta t \tag{25}$$

gives

$$\frac{Q^{(t+\delta t)}(a) - Q^{(t)}(a)}{\delta t} = \alpha[u_i^{(t)}(a) - Q^{(t)}(a)] \ , \tag{26}$$

and thus (12) follows by the definition of time derivative, $\dot{Q}^{(t)}(a)$, is the limit of the left hand side for $\delta t \to 0$.

A.3 Deriving (19)

Let us verify that (19) indeed satisfies (17):

$$\log p^{(t)}(a) = \beta u_i^{(t)}(a) - F \qquad \text{where } F = \log \sum_{a'} e^{\beta u_i^{(t)}(a')} \tag{27}$$

hence

$$u_i^{(t)}(a) - \sum_{a'} p^{(t)}(a') u_i^{(t)}(a') = \frac{1}{\beta}(F + \log p^{(t)}(a)) - \sum_{a'} p^{(t)}(a') \frac{1}{\beta}(F + \log p^{(t)}(a')) \quad (28)$$

$$= \frac{1}{\beta} \log p^{(t)}(a) - \sum_{a'} p^{(t)}(a') \frac{1}{\beta}(\log p^{(t)}(a')) \quad (29)$$

that is (17).