Limitations of Reinforcement Learning Algorithms in Imperfect Information Games

Jakub Koubele

1 Introduction

Reinforcement learning[1] (RL) is an area studying agents acting in an environment while maximizing some cumulative numerical reward. This problem is very often modeled as Markov Decision Process (MDP), and the RL task is thus to solve this MDP, i.e. to find a policy maximizing agent's cumulative reward. Some RL algorithms allow the agent to learn by direct interaction with the environment, without having knowledge of the underlying model.
Game theory[2] is an area studying interaction of multiple agents in games. These are environments where the reward of one agent can be affected by actions of other players. In such setting, the notation of optimality valid in single-agent environment becomes ambiguous, as we have separated rewards for each agent. Game theory therefore studies properties of some equilibria, which consist of agent's policies optimal simultaneously against each other. Compared to the RL task, finding equilibria in a game is usually a more complicated problem. That is because of some aspect specific to games that are not an issue in the RL setting: for example, agents may need to develop a probabilistic strategy to stay unpredictable for their opponents.
Despite these complications arising in games, RL algorithms were recently used to create a strong agents capable of beating professional human players in large games, such as DotA and Starcraft. This approach consist of RL agents playing the game against each other, while simultaneously improving their strategy just like they would be acting in an non-adversarely environment.
The aim of this thesis is to analyze the performance of RL algoritmhs that learn to play a game through self-play. Specifically, we will try to either provide a theoretical guarantees for these algorithms, or counter-examples of games where the convergence to equilibrium through self-play is hard or impossible. While game theory provide tools to analyze different multi-agent settings, we will focus only on so called two-players zero-sum games. These games can model environment with two players having completely opposite goals, thus trying to defeat each other.

2 Background

In this chapter, we will provide an overview of game theory and reinforcement learning.
We will begin by introducing the bandit problem, which consist of one agent sequentially choosing from a set of actions, while staying in the single state. Bandit problem is a precursor to full reinforcement learning setting, which adds multiple states and transitions between them. Special case of bandit problem, called adversarial bandits, have also linkage to the problem of finding an equilibrium of game through self-play.
We will continue by describing two-players, zero-sum normal form games. These consist of two agents choosing from their set of action; the reward of the players depends on the actions choosen by both of them. As in the bandit problem, normal form games do not include multiple states and transition between them.
TO-DO: Full RL
TO-DO: Extensive games

2.1 Bandit Problem

Bandit[1] problem is a simple model of sequential decision makeing with connections both to reinforcement learning and to normal form games.

Definition 2.1. A bandit problem with time horizont $T \in N$ is a tuple (A, X), where

- A $= \{1, ..., k\}$, $k \in N$ is a set of action available to the agent
- $X = \{x_{at}\}$ where $x_{at} \in R$ is a reward for playing an action $a \in A$ at a timestep $t \in \{1, ..., T\}$.

The agent in each round select an action a_t and then observe reward x_{ta}.

Based on the assumptions imposed on the reward distribution X, we can distinguish different categories of bandit problems. We will focus on two of them: stochastic bandits and adversarial bandits. Stochastic bandits serve as a precursor for the reinforcement learning problem, while adversarial bandits are relevant due to their linkage to solving a two-players zero-sum games.

2.1.1 Stochastic Bandits

In stochastic bandit problem, there exist probability distributions $p_a, a \in A$ such that $x_{at} \sim p_a \quad \forall t \in \{1, ..., T\}$. That is, reward for playing an action is i.i.d. sampled from a probability distribution (unknown to the agent), that depends only on the action played (and not on the timestep t). We also usually assume that the means of the distributions lie in some bounded interval and that their variance is finite. The task of the agent is to maximize his cumulative reward. To do so, agent need to learn which distribution provide the best reward. Since

[1] Name bandit came from some degens playing slots in Las Vegas.

he only observe the reward of the action he played, he need to do it by trail and error method. Compared to the task of supervised learning, the agent faces so called "exploration versus exploitation dilemma": he want to play actions that yield high payoff in his estimation, but simultaneously, he wants to improve his estimate of other actions (to be able to found these with high payoff.) Thus, agent may want to play action that is suboptimal by his current estimate, if the reward observed from playing that actions would provide enough information to help him decide whether or not is the action indeed suboptimal.
If we know more about the properties of distributions that generate the rewards, we can construct more efficient method to deal with the exploration versus exploitation problem.
We will now present two agents that deal with stochastic bandit problem, and don't make any additional assumption about the reward distribution. These methods are relevant for this thesis, as we will later on study their self-play convergence properties in games.

Action-value method with ϵ-greedy exploration

The action-value method holds an estimate of expected reward for playing each action. These estimates are simply empirical averages of rewards obtained from playing corresponding action. The action-value method also has an exploration parameter $\epsilon \in (0, 1)$. When deciding which action to play, action-value method choose to play the action with highest estimated reward with probability $1 - \epsilon$, and to select the action uniformly at random otherwise (with probability ϵ). After playing the action, the reward is obtained and the estimate of corresponding action is updated. As the estimate is a simple average of rewards, it can be efficiently updated online.
Exploration parameter ϵ ensures that each action has a probability at least $\frac{\epsilon}{k}$ to be played in each round. Therefore, with increasing timestep t, each action is played (in expectation) more times and the estimate of its average reward has lower variance, meaning that the agent has higher probability to correctly identify the optimal action.
However, exploring uniformly at random means that the agent has a probability $\frac{\epsilon}{k}$ to play an action which reward estimate is very low, even if the variance of that estimate is low as well. To avoid such behavior, we may choose ϵ to be a decreasing function of timestep t. Simultaneously, we would like to try each action an unbounded number of times (as time horizont T goes to $+\infty$), to avoid having a positive probability of failing to learn the optimal action. Formally, we want to have

$$\lim_{t \to \infty} \epsilon(t) = 0$$

and

$$\sum_{t=1}^{\infty} \epsilon(t) = \infty$$

Such exploration scheme is called "greedy in the limit with infinite exploration" (GLIE). If the agent follows such scheme, he is guaranteed (with probability 1) to obtain the optimal payoff on average, as $T \to \infty$. This does not mean that the whole action-value method with ϵ-greedy exploration is the best possible way to deal with bandit problem, as we may choose to use more sophisticated algorithms to obtain faster convergence rate. However, the value-action method with ϵ-greedy exploration illustrates the idea of learning the value function, which is a critical component of many reinforcement learning algorithms.

Gradient Bandit Algorithm

Instead of learning the value function and using it to select an action, we may try to learn the optimal policy directly. Example of this approach is a gradient bandit algorithm, which is a precursor to policy gradient method frequently used in reinforcement learning.
Gradient bandit algorithm select the action to play by sampling from its policy $\pi(w)$. That policy represents a probability distribution over the set of actions, and is parametrized by parameter vector w. Specifically, the gradient bandit algorithm uses softmax function to represent its policy, and the parameter vector w represent the parameters of softmax, which are a one scalar w_a for each action $a \in A$. The probability of playing an action a according to the softmax is defined as follows:

$$\pi_a(w) = \frac{\exp(w_a)}{\sum_{i \in A} \exp(w_i)}$$

The learning process is done by updating the weights w, such that they will put a high probability on playing actions with high rewards and low probability on actions with low rewards. To simplify the analysis, we will now assume for a moment that the agent observes the whole reward vector r_t after playing an action at time t, not just the reward r_{at} for the action he played.
In such setting, the agent can at each timestep t compute the expected reward

$$E\left[r_t | \pi(w_t)\right] = \sum_{a \in A} \pi_a(w_t) r_{at}$$

Then, he can perform the stochastic gradient ascent on the expected reward: compute the gradient

$$\nabla_t = \frac{\partial E\left[r_t | \pi(w_t)\right]}{\partial w_t}$$

and update

$$w_{t+1} = w_t + \eta_t \nabla_t$$

where η_t is a learning rate at time t.
TO-DO: pridat povidani o tom jak spravne diminishovat learning rate u SGD a jak to pak vede ke konvergenci.
We will now return to the setting where agent observes only the reward r_{at} for the action he played. In that setting, we need an unbiased estimator of the

whole reward vector r_t and use it instead of r_t. The only unbiased estimator in this setting is so called importance-weighted estimator, which works as follows: given a baseline $B \in R$, action a_t played at time t and observed reward for that action r, the estimate of r_{at} is computed as

$$\hat{r}_{at} = B + \frac{(r - B)I(a = a_t)}{\pi_a(w_t)}$$

where $I(a = a_t)$ is an indicator function equal to 1 if $a = a_t$ and 0 otherwise. The baseline B can be any number; we may choose it to be some constant, e.g. 0, or select it as some function of rewards observed in the past rounds. Even though that the selection of B will not affect the unbiasedness property of $\hat{r}_{at}$, it will affect its variance: we want thus to select B close to the observed reward r_t, in order to reduce the variance of the importance weighted estimator. The gradient bandit do it by choosing

$$B = \frac{1}{t}\sum_{t'=1}^{t} r_{t'}$$

that is, baseline at time t is an average of all rewards obtained up to time t.
If we replace the reward vector by its unbiased estimate given by the importance weighted estimator, the gradient bandit algorithm keeps all the convergence properties as in the full information setting.
TO-DO: mam trochu bordel ve znaceni rewardu

2.1.2 Adversarial Bandits

Kaslem na skoro vsechny assumptions jen z rewardy jsou bounded. Definice regretu (mozna poresit rozdil pseudoregretu a regretu). Relace k game theory: veta o tom ze no regret dynamika konverguje do epsilon NEQ, vlastni dukaz o lower boundu approximace. Mozna kratka uvaho o tom ze potrebujem avg. strategii, co je easy v tabular casu ale muze byt problem s function approximation, potrebovali bychom ukladat a pak evaluovat hodne neuronek coz neni dobry.
Nejaky povidani o policy na minimalizaci regretu, ze deterministicky policy nemuzou byt no regret a ze klicem je vhodny mixovani. Ukazat ze existuje Exp3, (mozna trochu diskutovat i Hedge predtim, protoze ho pak mozna budu srovnavat s tim policy grandient algoritmem). Ze Exp3 je skoro to co chceme jen ze ma velkou variance, kdyz to nejak poresime tam muzeme poustet (variantu) Exp3 v self-play a tim dosahnout (probabilistic-) garance konvergence do epsilon NEQ.

2.2 Normal Form Games

This section will introduce normal-form games and related solution concepts. The definitions are mostly taken from [3].

Definition 2.2. A 2-players zero-sum normal-form game is a tuple (N, A, u), where:

- $N = \{1, 2\}$ is a set of players,
- $A = A_1 \times A_2$, where A_1 (resp. A_2) is a finite set of actions available to player 1 (resp. player 2).
- $u = (u_1, u_2)$, where $u_i : A \to R$ is a payoff function for player i. It holds that $u_1(.) = -u_2(.)$.

The game is played as follows: both players choose an action from their sets of actions. Then, these actions are revealed and players obtain a payoff according to the payoff function. Since players need to choose an action without knowledge of the action chosen by their opponent, normal form games involve the aspect of imperfect information.
Payoff function of such game can be conveniently represented by a matrix, hence the normal-form games are also know as matrix games.
Even though the set of actions for each player is finite, players can decide for a probabilistic policy, choosing their actions randomly. Compared to a single-agent setting, this may be sometimes necessary condition for a policy to be optimal.

Definition 2.3. A set of mixed strategies for player $i = 1, 2$ is a $\Sigma_i = \Pi(A_i)$, where $\Pi(X)$ is a set of all probability distributions over set X.

We may consider deterministic policies to be a special case of mixed strategies:

Definition 2.4. We say that the strategy $\sigma_i \in \Sigma_i$ is pure, if σ_i plays some action with probability 1.

As the utilities of players depends on actions chosen by both of them, we will need following concept putting all their strategies together:

Definition 2.5. A strategy profile s is a Cartesian product of strategy sets of both players, $\Sigma_1 \times \Sigma_2$. We also define s_{-i} to be a strategy profile without agent i's strategy.

As a strategy profile induces a certain probability distribution over outcomes of the game, we may extend the concept of utility from actions to strategies and define

$$u(\sigma_1, \sigma_2) = E_{a_1 \sim \sigma_1, a_2 \sim \sigma_2}[u(a_1, a_2)]$$

Even though that the reward for playing some strategy depends on the strategy of the opponent, we may sometimes disregard certain strategies as 'bad', no matter what the opponent will do:

Definition 2.6. We say that strategy $\sigma_k \in \Sigma_i$ is weakly dominated by strategy $\sigma_l \in \Sigma_i$, if

$$u_i(\sigma_k, \sigma_{-i}) \leq u_i(\sigma_l, \sigma_{-i}) \quad \forall \sigma_{-i} \in \Sigma_{-i}$$

If the inequality is strict, we say that σ_k is strictly dominated. We may also speak about dominance of actions, if the strategies are pure.

Before we will define the concept of equilibrium, we will need the following concept of optimality from the point of view of one player:

Definition 2.7. Player i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in \Sigma_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$.

We may now define the concept of equilibrium:

Definition 2.8. A strategy profile σ^* is called an ϵ-Nash Equilibrium, if for each player i holds that

$$u_i(\sigma^*) + \epsilon \geq u_i(\sigma_i, \sigma^*_{-i}) \quad \forall \sigma_i \in \Sigma_i$$

where $\epsilon \in R$. If $\epsilon = 0$, we call σ^* just a Nash Equilibrium. In such case, each player i is playing best response against σ^*_{-i}.

Nash Equilibrium (NEQ) is the main solution concept of game theory. That is because rational players can be expected to play strategies according to NEQ - otherwise, at least one player would have an incentive to deviate from his current strategy. In the case of two-players zero-sum games, NEQ has another appealing property: playing according to NEQ is equivalent to maximize payoff against worst-case opponent's strategy. Thus, by playing strategy from NEQ, player i is guaranteed to obtain payoff at least $u_i(\sigma^*)$. The payoff $u_1(\sigma^*)$ is called the value of the game. In some games, the value of the game can be computed without solving the NEQ - for example, the value of symmetric game is 0. If we then compute ϵ-NEQ and play according to it, we are guaranteed to obtain at least some minimal payoff.

TO-DO: Sem nekam dat priklad s matching pennies a ze musime mixovat

Exact NEQ of two-players zero-sum normal form game can be found by solving a linear program which size is linear in the number of actions of players. However, we may also compute ϵ-NEQ through some learning processes, in which two agents play against each other and simultaneously improve their strategy. As this thesis try to analyze the performance of RL algortihms used to learn a game in such manner, we will now provide discussion of some learning dynamics which performance guarantees are well known.

2.3 Reinforcement learning

2.4 Extensive Form Games

3 Literature Review

- Obskurni clanky o gradient based metodach v 2x2 hrach. Mozna diskuze o tom z pro zero sum hry by to asi fungovalo pro hry n x n protoze to co tam delaj se asi da prevest na online convex optimization (jen by se musela brat avg. strategie misto current strategie)
- Smooth FP
- smooth Q-learning, Nash distribution
- FP v extensive form hrach, deep XFP
- Hard to read policy gradient od M. Lanctota a spol.

4 Self-play of RL Algorithms in Normal Form Games

4.1 Self-play of Action-value method with epsilon-greedy exploration

Chova se to jako FP jen to nema tolik infa. Obv to musime averagovat abychom meli mixovani. Musime jet GLIE jinak nam epsilon dava lower bound na konvergenci. Nemuzeme se zastavit v jinym stavu nez v pure NEQ. Ale i z pure NEQ muzeme vyskocit, a muzeme se naucit (na chvili) switchnout z dominovane strategie na dominovanou (kdyz u obou podhodnucejeme Q, kvuli nestejne rychlosti uceni kdyz mame jiny pocet pozorovani.)
Nasel jsem nejakou hru kde to moc nekonverguje, brutalni exhaustivni grid search parametru (nejprve epsilon-greedy, pak GLIE s racionalni funkci se dvema az trema parametrama).

4.2 Self-play of Gradient Bandit Algorithm

Chtelo by to vymyslet nejakej chytrej dukaz.

5 Sequential Setting

Tady uvarim nejakou neuronku ktera nebude konvergovat coz bude super. Asi to bude hrat poker, treba AKQ game s tabular metodama a pak neco co ma vic stavu (okolo 1k) s function approximation. Tahle cast bude dost meme.

References

[1] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Osborne, Martin J., and Ariel Rubinstein. *A course in game theory.* MIT press, 1994.

[3] Shoham, Yoav, and Kevin Leyton-Brown. Multiagent systems: *Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008

[4] Singh, Satinder, Michael Kearns, and Yishay Mansour. *Nash convergence of gradient dynamics in general-sum games.* Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 2000.

[5] Bowling, Michael, and Manuela Veloso. *Convergence of gradient dynamics with a variable learning rate.* ICML. 2001.

[6] Lattimore, Tor, and Csaba Szepesvári. *Bandit algorithms.* preprint (2018).

[7] Russell, Stuart J., and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[8] Waugh, Kevin. *Abstraction in large extensive games.* Diss. University of Alberta, 2009.