修　士　論　文

Neural Fictitious Self-Play を用いた不完全情報ゲームの均衡解の計算

Computing Equilibrium Strategies for Imperfect Information Games with Neural Fictitious Self-Play

指導教員　鶴岡 慶雅 教授

東京大学大学院工学系研究科
電気系工学専攻

氏 名　**37176431 河村 圭悟**

提 出 日　平成 31 年 1 月 25 日

概 要

二人零和不完全情報ゲームにおいて、ゲームの事前知識や状態遷移規則を知ることなくナッシュ均衡戦略を求めるタスクは重要である。しかしながら、既存の研究では大規模な不完全情報ゲームに対してナッシュ均衡戦略を求めようという研究は少なく、実際にそのようなゲームに適用した例も少ない。大規模な不完全情報ゲームに強化学習の手法を適用した例はあるものの、強化学習を用いた自己対戦は学習が収束すればこのナッシュ均衡戦略を得ることができる一方で、学習自体が収束することの証明は知られておらず、この手法が汎用的にナッシュ均衡戦略を得られるという保証はない。そこで本研究では、大規模な不完全情報ゲームにおけるナッシュ均衡戦略を安定して求められることが期待される、 Neural Fictitious Self-Play という手法について調べ、そのアルゴリズムの問題点と思われる部分を複数指摘し、その問題点を改善する手法を提案した。特に、その手法で用いられている強化学習の手法が、元来時間的に定常な環境で行う学習のために保持していた off-policy という有用な性質について、先述した手法と組み合わせることでその性質が失われてしまうことを指摘し、その事実を有効活用するための改善案として方策勾配法と呼ばれる手法が適用可能であることを示し、これを組み合わせた手法を提案した。また、この改善点を含む提案手法を、非自明な不完全情報ゲームである Mini-RTS というゲームに実際に適用し、得られたプレイヤを定量的・定性的に評価した。その結果、 Neural Fictitious Self-Play が方策勾配法と組み合わせて実際に非自明な不完全情報ゲームに適用可能であり、かつこの手法によって搾取されにくい戦略を得られることが確認された。

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

近年の深層学習の発展によって、人工知能（あるいは、機械学習）という単語が様々な場面で注目を浴びるようになっている。特に画像処理技術の分野においては、Google がニューラルネットワークによって猫を認識できるようになったというニュースを皮切りに、画像分類や物体認識、超解像、姿勢推定など、様々な分野の既存技術がニューラルネットワークを用いた機械学習によって取って代わられようとしている。これは主に畳み込みニューラルネットワークと動画像処理の相性が良く、人間が持つドメイン知識を適切にネットワークの構造に組み込めるからであるが、では機械学習は画像分野のみで発展しているのかというとそうではない。自然言語処理、すなわち日本語や英語などを適切に処理し、翻訳・要約などを行うタスクについても、ニューラルネットワークによる機械学習は目覚ましい成果を上げている。特にデータが豊富な機械翻訳については、従来の手法である統計的機械翻訳の精度を大きく上回っており、また転移学習が容易であるというニューラルネットワークの性質も相まって、日夜盛んに研究が行われている。

これらの分野に共通しているのは、時間的に定常なデータセットが存在しており、そのデータセットに様々な処理を行うことで予測や分類を行う、教師あり学習を用いているということである。ニューラルネットワーク（特に、誤差逆伝播法を用いた学習）は確率的勾配降下法によって目的関数を最大化・最小化するものであり、多くの場合この目的関数を、データセットからのサンプルで近似的に計算することで学習を行う。この学習方式を用いるために、ニューラルネットワークを用いた機械学習は本質的に教師あり学習へと帰着させることが多い。

しかしながら、解きたいタスクに対して常にデータセットがあるとは限らず、また分類・回帰のみでタスクが解けるとも限らない。いま、例として、ロボットのアームを適切に動かして物体を掴む、というタスクを考えよう。物体のデータセットと、各物体に対して適切に掴む操作列のデータセットがあれば、このタスクは教師あり学習に帰着させることができる。しかし、あり得る全ての物体について掴む操作列のデータセットを用意するのはコストが高く、また操作列を学習した場合、1 つ手順を間違えてしまった場合にその間違いをどう補填すればいいかを学ぶことができないため、失敗に対してロバストでないと考えられる。一方で人間がこのタスクを解く場合、データセットが与えられなくともそのロボットアームだけを渡してしまえば、しばらく操作した後に「コツ」を掴み、物体を掴めるようになるだろう。

このように、教師あり学習とは全く異なる枠組みで、データセットがなくとも試行錯誤を繰り返すことで学習を進める機械学習の分野として、強化学習というものがある。強化学習では、エージェントが環境と相互作用を繰り返し、得られたデータを元に行動の方針を更新することで学習を行う。エージェントには環境から報酬 (reward) が与えられ、この報酬を最大化するように行動の方針を更新する。先述のロボットアームの例では、エージェントはアームを何度も動かし、物体を上手く掴めた場合に正の報酬が貰えることによって、強化学習を通して少しづつ物体の掴み方を学んでいくことになるだろう。

この強化学習とニューラルネットワークを組み合わせたのが、深層強化学習と呼ばれる分野である。深層強化学習では、従来の強化学習のアルゴリズムのうち、状態に対するテーブルの部分をニューラルネットワークで近似することによって、環境のサイズに対するスケーラビリティを獲得している。また、ニューラルネットワークの汎化性能によって、到達したことのない状態についても適切に意思決定が行えるようになるという利点もある。深層強化学習のベンチマークとしてよく用いられるのが、Atari 2600 という 1 人用のビデオゲームである。Atari 2600 には多くのゲームが収録されているが、Mnih らは深層強化学習を用いることによって、同一のモデルとパラメータを用いて、人間の事前知識を一切用いることなく、画像入力のみから全てのゲームについて学習を行い、多くのゲームで人間のスコアを超えることに成功した [1]。この報告をきっかけに、様々な深層強化学習の手法が提案され、他のゲームについても同条件のもとで人間のスコアを超えられるようになってきている。

この Atari 2600 の例のように、強化学習が対象とする分野は主にエージェントが 1 人のゲームである。これは、強化学習のアルゴリズムの殆どが環境に MDP (Markov Decision Process, マルコフ決定過程) を仮定しているためである。MDP では、意思決定を行うエージェントは 1 人しかおらず、環境の状態遷移の確率分布は時間的に定常である。したがって、このモデルでは意思決定を行うエージェントが 2 人以上いるような状況、例えば対戦型のカードゲームや複数台のロボットの協調などは扱うことができない。

このような複数のエージェントがそれぞれ意思決定を行うようなタスクは、マルチエージェント (multiagent) タスクと呼ばれる。マルチエージェントタスクには協調型 (cooperative)、敵対型 (competitive) の 2 つと、それらを組み合わせた混合型 (mixed) がある。協調型は複数のエージェントに対して同一の報酬が返るタスクであり、目的はその報酬を最大化することである。一方、敵対型は複数（主に 2 人）のエージェントが相反する報酬を受け取るタスクであり、目的はそのゲームの解となる均衡戦略を求めることである。

本研究では、このうち敵対型のタスクを解くことを目指す。このタスクにおいては、お互いが自身の方針を切り替えることで得をしないような均衡点、ナッシュ均衡戦略を求めることが目的となる。この分野においても深層強化学習は多くの成果を上げており、昨年の夏には深層強化学習によって学習された AI が Dota 2 というリアルタイムストラテジーゲームにおいて人間のチームに勝利したという成果が報告された [2]。しかしながら、この分野には大きな問題点が残っている。強化学習を用いて自己対戦を行うことで、学習が収束すればこのナッシュ均衡戦略を得ることができる一方で、

強化学習による純粋な自己対戦では学習が収束することの証明は知られていない。したがって、得られたプレイヤは実験的に強いだけであり、汎用的にナッシュ均衡戦略を得られるという保証はどこにもないのである。

一方で、ゲーム理論の手法である Fictitious Play を改良し、ニューラルネットワークによる関数近似を用いるようにした Neural Fictitious Self-Play (NFSP) [3] という手法が提案されている。この手法は、他の強化学習の手法と同様に最小限の事前知識で学習が行える手法でありながら、元の Fictitious Play を近似する手法であり、大規模な不完全情報ゲームにおけるナッシュ均衡戦略を安定して求められることが期待される。しかしながら、強化学習による純粋な自己対戦と比べると学習が遅く、実用的ではないという面も存在している。

以上のことから本研究では、この NFSP について様々な面から研究を行い、性能を改善するための試行錯誤を行った。また、この手法を実際に大規模な不完全情報ゲームに適用し、得られたプレイヤを評価・観察することで、他の手法との比較や適用可能性についての考察を行った。その結果、NFSP が方策勾配法と組み合わせて実際に非自明な不完全情報ゲームに適用可能であり、かつ搾取されにくい戦略を得られることが確認された。

1.2 目的

本研究の目的は、敵対型のマルチエージェントタスク、あるいは二人零和不完全情報ゲームについて、事前知識などを用いることなくナッシュ均衡戦略を求める手法を模索することである。その上で、有用な手法であると考えられる NFSP について改善できる点や問題点などを指摘し、その修正を図り、一般の二人零和不完全情報ゲームを解くための手掛かりを作ることである。

1.3 貢献

本研究の貢献は以下の通りである。

- NFSP について、平均戦略の計算時に起こる対戦相手の戦略の変動を補正する手法（第 3.2 節）や、探索由来のデータを含めないことで教師あり学習の精度向上を図る手法（第 3.3）など、複数の観点から問題点を指摘し、その解決を図った。
- NFSP の強化学習が本質的に off-policy の特性を活かせないことを指摘し、その事実を有効活用する手段として方策勾配法を用いることを提案し、 方策勾配法と NFSP を組み合わせることが実際に可能であることを示した。

- 全探索不可能な不完全情報ゲームである Mini-RTS というゲームに提案手法を含む NFSP を適用し、通常の強化学習を用いた自己対戦の手法と定量的な比較を行うことで、通常の強化学習より学習は遅いものの、提案手法が実際に搾取されにくい戦略を獲得できることを示した。
- 学習が速く不安定であると考えられる自己対戦を用いて事前学習を行い、NFSP の学習速度を向上させる手法を提案した。また、実際にその手法が有効であるか実験を行い、ある程度の有用性を確認した。

1.4 構成

本稿の構成を以下に述べる。

まず、第 2 章では、本研究に関する前提知識について述べる。特に、第 2.1 節では、本研究の研究対象である展開型ゲームと、展開型ゲームと密接な関係にある標準型ゲームについて述べる。第 2.2 節では、本研究で求めたい対象であるゲームの解について述べる。第 2.3 節では、本研究で主に用いる手法である NFSP について述べる。第 2.4 節では、主に一人ゲームを解く手法である強化学習について、展開型ゲームと対応を取りながら述べる。本研究で用いる強化学習の手法についてもここで述べる。第 2.5 節では、本研究の比較実験に用いた、ゲームの木を探索することでナッシュ均衡戦略を効率良く得るゲーム理論の手法について、簡単に述べる。

第 3 章では、NFSP を改良するために行った様々な試行錯誤について述べる。第 3.1 節では、まずタスクを定式化し、その上でそのタスクに適用可能であると思われるいくつかの手法を比較し、本研究の最大の目的であるナッシュ均衡戦略を求めるにあたって NFSP が研究対象として妥当かを検討した。第 3.2 節では、 NFSP の平均戦略の計算時に起こる理論との乖離について指摘し、これを補正することで NFSP の性能が改善するかを検証した。第 3.3 節では、 NFSP の教師あり学習において探索由来の本来平均すべきでないデータが含まれてしまうことを指摘し、そのデータの除去を試みることで NFSP の性能が改善するかを検証した。

第 4 章では、実際に NFSP を非自明な不完全情報ゲームである Mini-RTS に適用し、 NFSP によって実験的に搾取されにくい戦略が得られるかを検証した結果について述べる。第 4.1 節では、手法を適用する対象である Real-Time Strategy ゲームの意義と難しさについて述べる。第 4.2 節では、NFSP の強化学習が本質的に off-policy ではないことを指摘し、その事実を有効活用する手段として方策勾配法を適用することを提案した。第 4.3 節では、 実際に提案手法を Mini-RTS に適用し、その結果について定量的・定性的な評価を行った。第 4.4 節では、前節で得られた考察から、自己対戦を用いて事前学習を行うことで NFSP の学習速度を向上させる手法を提案し、その手法を実際に適用して評価を行った。

第 5 章では、以上を踏まえて本研究を簡潔にまとめた。

第2章　前提知識

2.1　展開型ゲームと標準型ゲーム

本研究では、特定の不完全情報ゲームにとどまらず、できるだけ広範な分野について均衡解を求められる手法を模索し、その改善を図った。そのために、まず手法の適用対象となる有限展開型ゲームを定義する。また、展開型ゲームとは異なる形式である標準型ゲームを定義し、その 2 つの形式が表せるゲームの集合の差異について議論する。

2.1.1　展開型ゲーム

展開型ゲーム (extensive-form game) は、プレイヤの集合 $\mathcal{N}$ 、状態集合 $\mathcal{S}$ 、行動集合 $\mathcal{A}$ 、プレイヤ関数 P 、偶然手番の確率分布 F_c 、情報集合の集合 $\mathcal{U}$ 、情報集合を表す関数 I 、報酬関数 R のタプル $(\mathcal{N}, \mathcal{S}, \mathcal{A}, P, F_c, \mathcal{U}, I, R)$ からなる。どの集合も有限であるとき、このゲームを有限展開型ゲームと呼ぶ。

各状態 $s \in S$ について、対応するプレイヤ $i \in \mathcal{N} \cup \{c\}$ がプレイヤ関数 $P(s)$ によって決まる。ここで、 c は事前に確率分布 $F_c(s)$ によって定められた確率で行動を選択する偶然手番である。対応するプレイヤ i は、そのプレイヤにとっての観測である $I_i(s) \in \mathcal{U}_i$ を受け取り、行動 $a \in \mathcal{A}$ を選択する。 $I_i : \mathcal{S} \to \mathcal{U}_i$ は状態集合 $\mathcal{S}$ をそのプレイヤが区別できない状態の集合に分割する関数である。したがって、プレイヤ i は同じ情報集合 $u \in \mathcal{U}_i$ に属する状態 $s_1, s_2 \in u$ を区別できず、その 2 つの状態で全く同じ振る舞いをしなければならない。環境はプレイヤから行動 a を受け取り、状態を $s_{t+1} = (s_t a_t)$ と行動を繋げることで更新する。ゲームが終端状態 $\mathcal{Z}$ のいずれかに達したとき、各プレイヤ i には報酬関数 R_i に従って報酬が与えられる。報酬関数は終端状態に対して報酬の確率変数を返す関数である。

各プレイヤが各状態でどのように行動するかを定める確率分布を戦略 (strategy) と呼び、$\sigma \in \Sigma$ で表す。σ は情報集合 I に対して行動上の確率分布を返す関数である。各プレイヤの戦略 σ_i の組 $(\sigma_1, \cdots)$ を戦略プロファイルと呼ぶ。

2.1.2 標準型ゲーム

上述の定義では、状態 s は行動 a を繋げることで表現されていた。このように、展開型ゲームは状態をノード、行動をエッジとする木をなす。また、各プレイヤが過去の状態や行動を全て記憶しているとき、このゲームは完全記憶ゲーム (perfect recall game) であるという。完全記憶ゲームでは、状態だけでなく各プレイヤの情報集合も木になる。

展開型ゲームが状態の木でゲームを表現していたのに対し、標準型ゲームでは行動の組み合わせの表でゲームを表現する。標準型ゲーム (normal-form game) は、プレイヤの集合 $\mathcal{N}^N$ 、合法手集合 $\mathcal{A}^N$ 、報酬関数 R^N のタプル $\left(\mathcal{N}^N, \mathcal{A}^N, R^N\right)$ からなる。標準型ゲームでは各プレイヤ $i \in \mathcal{N}^N$ が同時に行動 $a_i \in \mathcal{A}_i^N$ を選択し、その組み合わせによって各プレイヤに報酬関数 R_i^N に従って報酬が与えられる。

2.1.3 展開型ゲームと標準型ゲームの相互変換

標準型ゲームにおいて、各プレイヤの行動を選択する確率分布を混合戦略 (mixed strategy) と呼び、$\sigma^N \in \Sigma^N$ で表す。混合と名がついているのは、標準型ゲームでは行動を純粋戦略 (pure strategy) と呼ぶこともあるためである。

実際、標準型ゲームにおける行動は展開型ゲームにおける純粋な、すなわち決定論的な (deterministic) 戦略を表している。ここでは以下のようにして有限展開型ゲームを有限標準型ゲームに変換する。

まず、有限展開型ゲームの各情報集合 $I \in \mathcal{U}$ と、その状態における各行動 $a \in \mathcal{A}$ の組 (I, a) を考える。この組の数は有限であるから、特に全ての情報集合についてそれぞれ行動を一つずつ選択した $\left(\left(I^1, a^1\right), \cdots, \left(I^{\#\{\mathcal{U}\}}, a^{\#\{\mathcal{U}\}}\right)\right)$ を考えてもよい。この、各情報集合についてそれぞれ行動を一つずつ選択したものが、標準型ゲームにおける純粋戦略に相当する。なぜなら、このように純粋戦略を定めることで、各プレイヤが同時に一つ純粋戦略を選べば各終端状態への到達確率が確定し（ただし偶然手番があるため終端状態は一意には定まらない)、従って報酬関数の確率分布も定めることができるからである。

元のゲームが完全記憶ゲームであるとき、この標準型ゲームにおける任意の混合戦略 $\sigma^N \in \Sigma^N$ に対し、その混合戦略と「等価」な戦略 $\sigma \in \Sigma$ が存在する [4]。ここで、戦略と混合戦略が等価であることを表すために、まず実現確率 (realization plan) を定義する。任意のプレイヤ $i \in \mathcal{N}$ と任意の情報集合 $I_i \in \mathcal{U}_i$ について、ゲームが完全記憶ゲームであれば、初期状態からその情報集合に到達するために必要な行動の列 $\boldsymbol{a} = (a_1, \cdots)$ がただひとつ存在する。この行動の列をプレイヤ i が取る確率は、プレイヤの戦略あるいは混合戦略が定まれば一意に定まる。この確率を情報集合 I_i の実現確率と呼ぶ。プレイヤ i の 2 つの戦略あるいは混合戦略について、任意の情報集合における実現確率が等しいとき、それらの戦略は実現等価 (realization equivalent) であるという。この意味で、Kuhn は

完全記憶ゲームにおける任意の混合戦略に対し、その混合戦略と実現等価な戦略が存在することを示した [4]。また、任意の戦略に対して、明らかにすべての行動の列について実現確率を実際に計算することができるから、その戦略と実現等価な混合戦略が存在することもわかる。

以上のことから、完全記憶展開型ゲームを考えるとき、その対応する標準型ゲームを考え、その上で求めた混合戦略を展開型ゲームにおける戦略に変換してもよい。標準型ゲームは状態の区別がないため、展開型ゲームより理論的な取り扱いがしやすいという利点がある。そのかわりに、展開型ゲームを標準型ゲームに変換する場合、標準型ゲームにおける行動の数が展開型ゲームの木の深さに対して指数的に増加するという欠点があり、実際のタスクに適用する場合にはスケーラビリティの問題が生じることもある。

2.2 均衡戦略

本研究の目的は、不完全情報ゲームの解を求める手法を模索することである。本節では、ゲームの解となる戦略について説明し、その観点から戦略を定量的に評価する方法についても説明する。

2.2.1 最適応答戦略

戦略プロファイル σ に対し、プレイヤ $i \in \mathcal{N}$ の得る報酬の期待値

$$r_i(\sigma) = \mathbb{E}\left[R_i(z)\right] \tag{2.1}$$

を計算することができる。ここで、期待値は各状態で行動を σ にしたがって選択したときの期待値である。これを戦略プロファイル σ におけるプレイヤ i の期待報酬と呼ぶ。プレイヤ i 以外の戦略 σ_{-i} を固定し、プレイヤ i の戦略を自由に動かしたとき、期待報酬 $r_i(\sigma_i, \sigma_{-i})$ を最大化するような戦略

$$\sigma_i^* = \arg\max_{\sigma_i \in \Sigma_i} \left[r_i(\sigma_i, \sigma_{-i})\right] \tag{2.2}$$

を考えることができる。この戦略 σ_i^* を、σ_{-i} に対する最適応答戦略 (best response strategy) と呼び、$\beta(\sigma_{-i})$ で表す。また、この最適応答戦略の期待報酬 $r_i(\beta(\sigma_{-i}), \sigma_{-i})$ を最適応答価値 (best response value) と呼び、 $b_i(\sigma_{-i})$ で表す。

期待報酬と最適応答価値の差が ϵ 以下になるような戦略のことを、σ_{-i} に対する ϵ-最適応答戦略と呼ぶ。最適応答戦略は 0-最適応答戦略である。

2.2.2 ナッシュ均衡戦略

展開型ゲームにおいて、戦略プロファイル σ の各戦略 σ_i がいずれも他のプレイヤへの最適応答戦略になっているとき、すなわちどのプレイヤも σ から戦略を切り替えることで期待報酬を増加させることができないとき、その戦略プロファイルはナッシュ均衡戦略 (Nash equilibria) であるという。すなわち、戦略プロファイル σ がナッシュ均衡であるとは、

$$\forall i \in \mathcal{N},\ \forall \sigma_i^* \in \Sigma_i,\ r_i(\sigma_i^*, \sigma_{-i}) \leq r_i(\sigma) \tag{2.3}$$

であることを言う。このとき、プレイヤ i の戦略 σ_i は σ_{-i} への最適応答戦略になっている。

ナッシュ均衡戦略においては、各プレイヤが（相手プレイヤを固定して）自身の戦略を変更する動機をもたないので、例えば各プレイヤが順番に自身の戦略をアップデートしていくような状況下においては、ナッシュ均衡戦略はそのアルゴリズムの均衡点になる。

また、ゲームが二人零和であるとき、ナッシュ均衡戦略はあらゆる戦略に対する期待報酬の最小値を最大化するような戦略になる。すなわち、戦略 σ^* がナッシュ均衡戦略であるとき、

$$\forall i \in \mathcal{N},\ \min_{\sigma_{-i} \in \Sigma_{-i}} r_i(\sigma_i^*, \sigma_{-i}) = \max_{\sigma_i \in \Sigma_i} \min_{\sigma_{-i} \in \Sigma_{-i}} r_i(\sigma_i, \sigma_{-i}) \tag{2.4}$$

が成り立つ。言い換えれば、ゲームが二人零和でありかつ対称ゲームであれば、ナッシュ均衡戦略は「あらゆる戦略に対して負けない戦略である」ということができる。この意味で、ナッシュ均衡戦略をゲームの解と呼ぶことがある。本研究の目的は、このゲームの解を求めることである。

2.2.3 可搾取量

戦略がナッシュ均衡戦略であるとき、その戦略はあらゆる戦略に搾取されない戦略である。この意味で、戦略がどの程度搾取されやすいかを表す値を定めれば、その戦略がどれだけナッシュ均衡戦略に近いかを定量的に評価することができる。

戦略の搾取されやすさは、あらゆる戦略と対戦させたときの相手の期待報酬の上限であると言える。この考えに基づいて、二人零和ゲームにおける可搾取量 (exploitability) $\epsilon(\sigma)$ は次式で定義される [5]。

$$\epsilon(\sigma) = \sum_{i \in \mathcal{N}} \max_{\sigma_i^* \in \Sigma_i} \left[r_i\left(\sigma_i^*, \sigma_{-i}\right) \right] = \sum_{i \in \mathcal{N}} b_i(\sigma_{-i}) \tag{2.5}$$

この式は、三人以上の零和ゲームについてもそのまま拡張できる [6, 7]。この値が 0 に近いほど、その戦略はナッシュ均衡に近いと言える。実際、任意の戦略 σ について

$$\epsilon(\sigma) = \sum_{i \in \mathcal{N}} b_i(\sigma_{-i}) \geq \sum_{i \in \mathcal{N}} r_i(\sigma) = 0 \tag{2.6}$$

であり、この等号が成り立つのはすべての $i \in \mathcal{N}$ について $b_i(\sigma_{-i}) = r_i(\sigma)$ が成り立つとき、すなわち σ がナッシュ均衡戦略であるときのみである。可搾取量が ϵ であるような戦略を、ϵ-ナッシュ均衡戦略と呼ぶ。ϵ-ナッシュ均衡戦略では、各戦略は相手への ϵ-最適応答戦略になっている。

2.3 Neural Fictitious Self-Play

本研究では、不完全情報ゲームのナッシュ均衡戦略を求めるための手法として、主に Neural Fictitious Self-Play (NFSP) [3] を用いた。この手法はゲーム理論のアルゴリズムである Fictitious Play (FP) [8] を基にしており、このアルゴリズムを展開型ゲームにも拡張し、さらにニューラルネットワークによる関数近似を適用したのが NFSP である。本節では、この NFSP のアルゴリズムについて概説する。

2.3.1 Fictitious Play

標準型ゲームにおいてナッシュ均衡戦略を求める手法に、Fictitious Play (FP) [8] がある。

ステップ t における混合戦略を $\sigma^{N,t}$ とする。いま、ステップ 1〜T までの混合戦略の平均

$$\pi^{N,T} = \frac{1}{T} \sum_{t=1}^{T} \sigma^{N,t} \tag{3.1}$$

を考える。混合戦略は確率分布であるから、和が再び 1 になるこの平均もまた混合戦略である。この平均戦略に対する最適応答戦略を

$$\beta_i^N \left(\pi_{-i}^{N,T} \right) = \max_{\sigma_i^N} u_i^N \left(\sigma_i^N, \pi_{-i}^{N,T} \right) \tag{3.2}$$

とする。この最適応答戦略を平均することを考えると、$\pi_i^{N,T}$ は

$$\pi_i^{N,T+1} = \frac{1}{T+1} \sum_{t=1}^{T+1} \sigma^{N,t} = \frac{1}{T+1} \beta_i^N \left(\pi_{-i}^{N,T} \right) + \left(1 - \frac{1}{T+1} \right) \pi_i^{N,T} \tag{3.3}$$

と漸化式の形で表すことが出来る。この更新式に従って平均戦略 $\pi^{N,T}$ を更新していくのが FP である。FP によって得られる平均戦略 σ_T^N は、2 人ゲームやポテンシャルゲームなど、いくつかのゲームにおいてナッシュ均衡解に収束することが証明されている。

また、これを一般化した手法である Generalized Weakened Fictitious Play (GWFP) [9] では、式 (3.3) を以下のように拡張する。

$$\pi_i^{N,T+1} = \alpha^{N,T}\left(\beta_{i,\varepsilon^T}^N\left(\pi_{-i}^{N,T}\right) + M_T^N\right) + \left(1-\alpha^{N,T}\right)\pi_i^{N,T} \tag{3.4}$$

ここで、$\beta_{i,\varepsilon}^N$ は ε-最適応答戦略を表す。$\alpha^{N,T}$、ε^T はともに $T \to \infty$ で 0 に収束し、$\sum_t^\infty \alpha^{N,t} = \infty$ を満たす任意の数列である。M_T^N は摂動項であり、$\forall A > 0$ について以下の条件式を満たす数列である。

$$\lim_{T\to\infty}\sup_K\left\{\left\|\sum_{t=T+1}^K \alpha^{N,t}M_t^N\right\| \text{ s.t. } \sum_{t=T+1}^K \alpha_t^N \le A\right\} = 0 \tag{3.5}$$

式 (3.4) において、$\alpha_T^N = 1/(T+1)$、$\varepsilon_T = M_T^N = 0$ とすればこれは FP になる。

GWFP によって得られる平均戦略 σ_T^N は、FP 同様、いくつかのゲームにおいてナッシュ均衡解に収束することが証明されている。

2.3.2 Extensive-Form Fictitious Play

FP は、標準型ゲームにおけるナッシュ均衡戦略を求める手法であった。標準型ゲームは状態を持たず、各プレイヤが行動を一度だけ選択することでゲームが終了する。しかし、現実の問題やより複雑なゲームは、行動を選択することで内部状態が遷移し、再び行動を選択する、という展開型ゲームの形で記述されることが多い。先述の通り、展開型ゲームにおける戦略と実現等価な標準型ゲームの混合戦略があるから、展開型ゲームを一度標準型ゲームに置き換え、その上でゲームを解き、再び展開型ゲームに戻すという操作を行えば、標準型ゲームにおけるナッシュ均衡戦略を求める手法でも、展開型ゲームにおけるナッシュ均衡戦略を求めることができる。しかし、この変換の過程で展開型ゲームにおけるすべての可能性を考慮した行動を作成しなければならないため、標準型ゲームにおける行動の数は展開型ゲームの木の深さに対して指数的に増加する。したがってこの方法は現実的ではない。

これに対して Heinrich らは、展開型ゲームに対してゲーム木の深さについて高々多項式程度の計算量で FP を行う EXtensive-form Fictitious Play (XFP) と呼ばれる手法を提案した [10]。XFP では、標準型ゲームにおける平均戦略の計算、すなわち混合戦略の加法を、展開型ゲームにおける戦略の加法で表せるようにすることで、標準型ゲームに落とし込むことなく FP の計算を実現している。

第 2.1.3 項と同様に、実現確率、および実現等価性を定義する。また、情報集合 $I_i \in \mathcal{I}_i$ に対し、戦略（あるいは混合戦略） π_i を持つプレイヤ i の、情報集合 I_i の実現確率を $x_{\pi_i}(I_i)$ とする。このとき、以下の定理が成り立つ。

2 つの戦略 π, β それぞれに対して実現等価な混合戦略 π^N, β^N と、$\lambda_1, \lambda_2 \le 0, \lambda_1 + \lambda_2 = 1$ を満た

す実数の組 λ_1, λ_2 について、

$$\mu(I,a) = \frac{\lambda_1 x_\beta(I)}{\lambda_1 x_\pi(I) + \lambda_2 x_\beta(I)}\pi(I,a) + \frac{\lambda_2 x_\beta(I)}{\lambda_1 x_\pi(I) + \lambda_2 x_\beta(I)}\beta(I,a) \tag{3.6}$$

で表される戦略 μ は、混合戦略 $M = \lambda_1 \pi^N + \lambda_2 \beta^N$ と実現等価である。

この定理を直感的に理解するためには、 $x_\pi(I)$ が 0 や 1 であるようなときを考えるとよい。実現確率が 0 であるとき、その情報集合における局所的な戦略は通常の確率分布であるが、実際にその情報集合へ到達することはない。したがってこのような戦略は加法に一切影響を与えない。逆に、その情報集合の実現確率が 1 であるとき、その情報集合を含まない部分木はすべて無意味であるから、通常の混合戦略と同様に足し引きを行っても構わない。

この定理によって、標準型ゲームにおける混合戦略の加法を、展開型ゲームにおける戦略の加法で表すことができる。すなわち、第 2.3.1 節で述べた GWFP は

$$\pi_i^{t+1}(I,a) \propto \alpha^t x_{\beta_i^{t'}}(I)\,\beta_i^{t'}(I,a) + \left(1-\alpha^t\right) x_{\pi_i^t}(I)\,\pi_i^t \tag{3.7}$$

と表現できる。ただし、$\beta_i^{t'}(I,a) = \left(\beta_{i,\varepsilon^t}\left(\pi_{-i}^t\right) + M_t\right)(I,a)$ とした。この計算はゲーム木の深さに対して高々多項式程度の計算量なので、展開型ゲームで表されるゲームについても効率よくナッシュ均衡解を求めることができる。これが XFP の考え方である。

2.3.3 Fictitious Self-Play

前項で述べたように、XFP では展開型ゲームにおける戦略の加法を考えることで、標準型ゲームを介することなく FP のアルゴリズムを実現した。Heinrich らはさらに、この XFP に機械学習の手法を取り入れ、関数近似を行った Fictitious Self-Play (FSP) という手法を提案した [10]。この手法では、 XFP における最適応答戦略の計算を強化学習で、平均戦略の計算を教師あり学習で近似することで、機械学習の最新の手法を取り入れつつ、ゲームの状態空間の大きさに対してスケールするようになっている。

展開型ゲームと戦略プロファイル π を考える。いま、あるプレイヤ i について、それ以外のプレイヤの戦略 π_{-i} を固定した 1 人ゲームを考えると、このゲームは後述する Markov Decision Process (MDP, マルコフ決定過程) をなす [11]。この MDP に対して強化学習の手法を用いることで、戦略 π_{-i} に対する最適応答戦略を求めることができる。強化学習で厳密な最適応答戦略を求めるには当然多くの時間が掛かるが、 FP においては各プレイヤの戦略は平均戦略に従うため、時刻 t における戦略 π_{-i}^t と時刻 $t+1$ における戦略 π_{-i}^{t+1} の差は学習が進むにつれて小さくなっていくと考えられる。実際、戦略が FP に従って更新されるとき、時刻 t における平均戦略 π_{-i}^t への ϵ_t-最適応答戦略 β_{i,ϵ_t}^t を考えると、この戦略は時刻 $t+1$ における平均戦略に対する $\left[\epsilon_t + \frac{1}{t+1}(R-\epsilon_t)\right]$-最適応答戦略にもなっている [10]。ここで、$R$ はあらゆる戦略を考えたときの期待報酬の最大値と最小値の

差である。したがって、 FP のステップを繰り返しながら強化学習を少しずつ行うことで、効率よく FP の手順を行うことができると考えられる。

また、平均戦略の計算に関しては、自身の時刻 t における（近似的な）最適応答戦略 β_i^t と固定された戦略 μ_{-i} の組 (β_i^t, μ_{-i}) を考え、 $\left(\beta_i^1, \mu_{-i}\right), \cdots, \left(\beta_i^T, \mu_{-i}\right)$ から等確率で状態と行動をサンプリングすれば、そのサンプルを再現するような分布は平均戦略 π_i^T になる。したがって、強化学習で得られた各時刻の戦略から行動をサンプリングし、そのデータに対して教師あり学習を行うことで、FP における平均戦略を教師あり学習で再現することができる。この際、相手の戦略 μ_{-i} は時刻に対して固定されていなければならないが、 FP における平均戦略は時刻に対して十分遅く変化すると考えられるため、相手の戦略に π_i^t を用いてもよい、と Heinrich らは主張している [10]。

2.3.4 Neural Fictitious Self-Play

前項で述べた FSP では、 FP に機械学習の手法を用いて関数近似を導入することで、 FP の収束の保証を保ちながらスケーラビリティを獲得していた。 Neural Fictitious Self-Play (NFSP) [3] は、この FSP にニューラルネットワークを組み合わせた手法である。

NFSP では、強化学習の手法として後述する Deep Q-Network (DQN) を用いる。また、教師あり学習の平均戦略の計算に、reservoir replay buffer を用いる。さらに、教師あり学習と強化学習を同時に行うために、 anticipatory parameter を導入し、平均戦略の予測を行う。本項では、後述する DQN の説明は省略し、 reservoir replay buffer 、および anticipatory parameter の説明を行う。

2.3.4.1 Reservoir Sampling

FSP の教師あり学習において、教師あり学習で学習する対象となる戦略は（実現等価な混合戦略上で）式 (3.3) のようになる。すなわち、

$$\pi_i^T = \left(1 - \frac{1}{T}\right) \pi_i^{T-1} \oplus \frac{1}{T} \beta_i^T \left(\pi_{-i}^{T-1}\right) \tag{3.8}$$

となる。ここで、$\oplus$ は式 (2.1) で表される戦略上の加法である。

この式を見るとわかるように、 FSP の学習を回すためには、各時刻 t において過去の戦略 π_i^{T-1} から何度もサンプリングを行う必要がある。しかしながら、教師あり学習の更新はサンプリングによって行われるために、この手順を繰り返すと過去の戦略の記憶が劣化し、正しく平均を計算することができないと考えられる。また、新たに学習されているのは最適応答戦略の β_i^T の部分であり、この戦略からデータがサンプルされる確率 $\frac{1}{T}$ が時刻に対して減衰するのは効率が悪いと考えられる。

NFSP では、上の式に従って平均戦略の計算を行う代わりに、過去の強化学習の戦略 β_i^t から得られたデータを reservoir replay buffer に保存し、その buffer 内の分布を再現するように教師あり学

習を行う。Reservoir replay buffer は、 この直後に述べる reservoir sampling を用いた buffer であり、これによって過去の $\beta_i^1, \cdots, \beta_i^T$ から等確率にデータをサンプルして保存することができる。これにより、教師あり学習が保持している過去の戦略を参照しなくても平均戦略の計算が行えるので、学習が安定化すると考えられる。

Reservoir sampling [12] は、総数が分からないデータの組から、与えられた数のデータを等確率に選択する手法である。

N 個のデータ $d_1, d_2, \cdots, d_N$ から、ランダムに $k(< N)$ 個を選択し、メモリ M に格納することを考える。N が分かっており、$d_1, d_2, \cdots, d_N$ が全て与えられている場合は、単にデータを 1 つ等確率に選択して M に格納し、いま選択したデータを除く、という操作を繰り返せばよい。しかしながら、総数 N が分かっておらず、データが d_1, d_2 と順に与えられる場合、データを等確率に選択することができないため、この操作を行うことはできない。また、N が十分に大きく、データ $d_1, d_2, \cdots, d_N$ をメモリに格納できない場合であっても、この操作を行うことはできない。

reservoir sampling は、次のようにして行われる。まず、k 個のデータを格納できるメモリ $M = \{m_1, m_2, \cdots, m_k\}$ を用意する。与えられたデータ d_i について、$i \leq k$ であれば、$m_i \leftarrow d_i$ とする。すなわち、最初の k 個のデータは一度すべて選択する。もし $i > k$ であれば、1 以上 i 以下の整数乱数を発生させ、その値を r とする。$r > k$ であればそのデータは選択しない。$r \leq k$ であれば $m_r \leftarrow d_i$ とする。すなわち、$k+1$ 個目以降新たに来た i 番目のデータは、k/i の確率でメモリに格納され、$1 - k/i$ の確率で捨てられる。この操作を d_N まで繰り返したときのメモリ M は、$d_1, d_2, \cdots, d_N$ から等確率に k 個をサンプルしたものになっている。このアルゴリズムを、アルゴリズム 1 に示す。

reservoir sampling の主張が成り立っていることを、数学的帰納法を用いて示す。d_k までが与えられた時点で、メモリ M は全ての $1 \leq i \leq k$ について $m_i = d_k$ となっている。すなわち、k 個のデータ $d_1, d_2, \cdots, d_k$ から等確率に k 個を選択していると言える。いま、d_l までが与えられた時点で、メモリ M が d_l までの l 個のデータから等確率に k 個を選択していると仮定する。新たに d_{l+1} が与えられると、reservoir sampling によってこのデータは

$$\frac{k}{l+1} \tag{3.9}$$

の確率でメモリに格納される。一方で、それ以前に与えられたデータ $d_i(1 \leq i \leq l)$ は、仮定より k/l の確率でメモリに格納されているが、$(k/(l+1)) \times (1/k) = 1/(l+1)$ の確率で新たなデータ d_{l+1} によって上書きされてしまうので、この操作の後に d_i がメモリに格納されている確率は

$$\frac{k}{l} \times \left(1 - \frac{1}{l+1}\right) = \frac{k}{l+1} \tag{3.10}$$

となる。式 (3.9)、式 (3.10) より、d_l までが与えられた時点でメモリ M が等確率に k 個を選択していると仮定すると、reservoir sampling によって d_{l+1} が与えられた後でもメモリ M が等確率に k 個を選択していることが分かる。以上より、任意の $N \geq k$ について、reservoir sampling を用いて等確

アルゴリズム 1 Reservoir Sampling

Require:

データ $d_1, d_2, \cdots, d_N$、データを格納するメモリ M

Ensure:

M は $d_1, d_2, \cdots, d_N$ からランダムに k 個選択したもの

```
function RESERVOIR
    for i = 1, 2, ···, N do                    ▷ N が分かっていなくてもよい
        Func(i, d_i, k)
    end for
end function
function FUNC(i, d_i, k)
    if i ≤ k then                              ▷ 最初の k 個は無条件で格納する
        M_i ← d_i
    else
        r ← random(1, i)
        if r ≤ k then                          ▷ 確率 k/i で選択する
            M_r ← d_r                          ▷ ランダムな位置に格納
        end if
    end if
end function
```

率に k 個のデータを選択できることが示された。

2.3.4.2 Anticipatory Dynamics

FSP の学習では、各プレイヤが教師あり学習 π と強化学習 β の 2 つの戦略を持ち、かつそれぞれの戦略を更新することで学習が行われる。あるプレイヤ i に着目したとき、プレイヤ i の教師あり学習 π_i は、自身の強化学習の戦略 β_i^t からデータをサンプルし、そのデータを予測するように学習を行う。また、プレイヤ i の強化学習 β_i は、相手の教師あり学習の戦略 π_{-i} との対戦からデータをサンプルし、その期待報酬を最大化するように学習を行う。従って、プレイヤ i から見れば、自身は強化学習 β_i に従って行動し、対戦相手は教師あり学習 π_{-i} に従って行動するのが望ましい。

しかしながら、 NFSP では各プレイヤが同時に学習を行う。これを実現するために、 NFSP では anticipatory dynamics [13] を用いる。

教師あり学習の戦略 π_i は、式 (3.8) のように 1 ステップ前の自身の戦略に強化学習の戦略を足し合わせたものになる。これが連続的に変化するとすれば、教師あり学習の戦略は形式的に

$$
\begin{aligned}
\pi_i^T &= \left(1 - \frac{1}{T}\right) \pi_i^{T-1} \oplus \frac{1}{T} \beta_i^T \left(\pi_{-i}^{T-1}\right) \\
\pi_i^T \ominus \pi_i^{T-1} &= \frac{1}{T} \left(\beta_i^T \left(\pi_{-i}^{T-1}\right) \ominus \pi_i^{T-1}\right) \\
\frac{d}{dT} \pi_i^T &\propto \frac{1}{T} \left(\beta_i^T \left(\pi_{-i}^{T-1}\right) \ominus \pi_i^{T-1}\right)
\end{aligned}
\tag{3.11}
$$

と書くことができる。ゆえに、戦略

$$
\sigma_i^t = \left(1 - \eta^t\right) \pi_i^t \oplus \eta^t \beta_i^t = \pi_i^t + \eta^t \frac{1}{t} \left(\beta_i^t \ominus \pi_i^{t-1}\right) \simeq \pi_i^t + \eta^{t'} \frac{d}{dt} \pi_i^t \tag{3.12}
$$

を考えれば、これは教師あり学習の戦略 π_i^t の先読みを行った戦略であると考えることもできる。この η を anticipatory parameter と呼ぶ。さらに、この戦略 σ を各プレイヤが用いることで、強化学習は相手の教師あり学習の先読み戦略 σ_{-i} を相手とする学習ができ、教師あり学習は単に β_i^t からデータをサンプルしたときだけ学習を行えばよいので、各プレイヤを同時に学習させることができる。

以上をまとめた NFSP のアルゴリズムを、アルゴリズム 2 に示す。

2.4 強化学習

強化学習は、Markov Decision Process (MDP) という比較的緩い制約のもとで最適な戦略を求めるための機械学習の手法である。 MDP では意思決定を行うプレイヤは 1 人しかいないが、対戦相手を固定することで、あるいは対戦相手の戦略の変化が十分遅いという仮定を置くことで、多人数ゲームにも適用できるようになる。また、本研究では、ゲーム理論と機械学習を組み合わせた手法を研究対象としており、その手法にも強化学習の技術が用いられている。これらのことから、本節では強化学習について必要な知識を述べる。

2.4.1 Markov Decision Process

Markov Decision Process (MDP、マルコフ決定過程) は、プレイヤが意思決定を行いながら環境と相互作用し、報酬を得るという枠組みである。

MDP は以下の要素からなる。

- 状態 s の有限集合 $\mathcal{S}$。
- 状態 $s \in \mathcal{S}$ で取ることが出来る行動の集合 $\mathcal{A}_s$。

- 遷移関数 $T(s,a,s')$ 。 $T(s,a,s')$ は状態 $s \in \mathcal{S}$ において行動 $a \in \mathcal{A}_s$ を取ったとき状態 $s' \in S$ に遷移する確率である。
- 報酬関数 $R(s,a,s')$。$R(s,a,s')$ は状態 $s \in S$ において行動 $a \in \mathcal{A}_s$ を取り、状態が $s' \in S$ に遷移したとき環境から与えられる報酬を表す確率変数である。
- 初期状態 $s_0 \in \mathcal{S}$。

さらに、プレイヤが各状態でどのように行動するかを定める確率分布を、展開型ゲームと同様に戦略と呼び、π で表す。π は状態 s に対して行動上の確率分布を返す関数である。

なお、 MDP においては意思決定を行うプレイヤをエージェント (agent) 、行動上の確率分布を方策 (policy) と呼ぶことが一般的であるが、本稿ではゲーム理論の文脈に合わせてそれぞれプレイヤ (player) 、戦略 (strategy) と呼ぶ。同様に、展開型ゲームにおいては行動の列を繋げたゲームの内部状態を履歴 (history) と呼ぶことが一般的であるが、本稿ではこれを状態 (state) と呼ぶ。

MDP におけるプレイヤの目的は、期待累積報酬

$$\mathbb{E}\left[\sum_t \gamma^t R(s,a,s')\right] \tag{4.1}$$

を最大化することである。ここで、γ は $0 < \gamma \leq 1$ を満たす実数で、エピソードが有限でないときに累積報酬を収束させるために、あるいは今の状態に近い報酬により高い価値を付けるために導入される値であり、割引率 (discount factor) と呼ばれる。期待累積報酬を最大化するような戦略を最適戦略と呼ぶ。 MDP における強化学習では、状態 s と報酬 r だけを観測しながら、最適戦略を求めることが目的となる。

MDP の特徴は、状態の遷移関数が一つ前の状態とプレイヤの行動にしか依存しないことである。現実の問題においては、状態遷移はプレイヤの観測できる状態以外にも依存している場合が多く、言い換えればプレイヤに与えられる観測とゲームの状態が異なっている。そのような設定を Partially Observed MDP (POMDP、部分観測マルコフ決定過程) と呼ぶ。 POMDP における最適戦略を一般に求めることは難しく、これを不完全情報ゲームの手法を用いて解く研究も存在しているが [14]、本稿では取り扱わない。

2.4.2 Q 学習

MDP の最適戦略を求める手法の一つに、Q 学習 (Q-learning) がある。Q 学習は、 MDP に対して次式で定義される最適行動価値関数

$$Q^*(s_t,a_t) = \max_\pi \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\right] \tag{4.2}$$

を予測するように学習を行う。ここで、r_t はステップ t における報酬である。

最適行動価値関数は、以下の方程式

$$Q(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim T(s_t, a_t)} \left[R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1} \in \mathcal{A}_{s_{t+1}}} Q(s_{t+1}, a_{t+1}) \right] \tag{4.3}$$

を満たす。この方程式をベルマン方程式 (Bellman equation) と呼ぶ。逆に言えば、この方程式を満たすように関数を更新していけば、最適行動価値関数が求まるのではないかと考えられるだろう。実際、Q 学習では関数 $Q(s, a)$ がこの方程式を満たすように学習を行う。すなわち、状態 s とアクション a について関数 $Q(s, a)$ を考え、これを次のような更新式に従って更新する。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left(R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1} \in A_{s_{t+1}}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \tag{4.4}$$

ここで、α_t は学習率 (learning rate) である。これは式 (4.3) のサンプルしたデータにおける二乗誤差を勾配降下法で最小化するのと同じことである。

学習時には、プレイヤは行動戦略 (behavior strategy) β を持つ。この戦略として ε-greedy が用いられることが多い。ε-greedy は、確率 ε でランダムな行動を選択し、 $1 - \varepsilon$ で最良と思われる行動を選択する手法である。このとき、最良と思われる行動としては、単に $Q(s, a)$ が最大になるような a を選択する。$Q(s, a)$ が最適行動価値関数 $Q^*(s, a)$ に十分近ければ、$Q(s, a)$ は今の状態 s から行動 a を選択し、その後最良の行動を取り続けたときの期待累積報酬を表すからである。

ε-greedy を用いてランダムな行動を行う理由は、最良と思われる行動だけを取り続けているとまだ見ていない高い報酬に出会うことができず、局所的な解に落ちてしまう可能性があるからである。このように、局所解に落ちないように最良ではないと思われる行動も取ることを、探索 (exploration) と呼ぶ。逆に、最良と思われる行動を取ることを活用 (exploitation) と呼ぶ。探索と活用のジレンマは、Q 学習だけでなく強化学習一般に起こる問題である。

また、テスト時には、プレイヤは学習対象の戦略 (target strategy) π に従って行動する。この戦略は、Q 学習においては $Q(s, a)$ が最大になる a を確率 1 で選択する戦略である。後述する通り、このように行動戦略と学習対象の戦略に別々の戦略を用いることができるのが Q 学習のような value-based な手法の利点の一つである。

Q 学習は、学習率 α_t が $\sum_t \alpha_t = \infty$ かつ $\sum_t \alpha_t^2 < \infty$ を満たすとき、 $Q(s, a)$ が最適行動価値関数 $Q^*(s, a)$ に収束することが証明されている [15]。

2.4.3 Deep Q-Network

Mnih らは、Q 学習にニューラルネットワークを組み合わせた Deep Q-Network (DQN) を提案した [1]。DQN では、前述した Q 関数をニューラルネットワークで関数近似し、その関数がベルマン

方程式を満たすように学習を行う。また、時間的に非定常なデータをニューラルネットワークで扱う上で生じる不安定性を解消するために、 replay buffer や target network などの工夫を行っている。

DQN における学習は、以下のようにして行われる。まず、通常の Q 学習と同じように、プレイヤは ε-greedy で行動し、データ (s_t, a_t, s_{t+1}, r_t) を集める。集まったデータは circular replay buffer に格納される。 Circular replay buffer は、過去のデータを貯めておく buffer であり、データが溢れた場合は古い方から順に捨てることでデータを保持する。この buffer からデータをランダムに取り出し、そのデータについて以下の損失関数

$$L = \sum \left(r_t + \gamma \max_{a \in A_{s_{t+1}}} Q(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta) \right)^2 \tag{4.5}$$

を最小化するように学習を行う。ここで、θ は Q 関数を表すニューラルネットワークのパラメータであり、 θ^- は target network のパラメータである。

ニューラルネットワークを用いている以外の点で DQN が従来の Q 学習と異なるのは、 replay buffer を使っている点と、 target network を使っている点である。前者を用いている理由は、単に探索で得られたデータを用いて学習を行った場合、1 ステップ前のデータの状態 s と、新たに得られたデータの状態 s' の相関が非常に高くなってしまい、学習が不安定になるためである。通常のニューラルネットワークの学習では、学習データが時間的に定常であるため、データを事前にシャッフルすることができ、データ内の（順番による）相関が問題になることは少ない。しかし、強化学習ではプレイヤが環境と相互作用しながらデータを収集するため、学習データを事前にシャッフルすることができない。これを解決するために、replay buffer が用いられている。また、後者を用いている理由は、式 (4.5) の θ^- を θ に置き換えた場合（微分グラフは切るものとする）、学習の損失が自己依存になってしまい、学習が不安定になることが実験的に知られているからである。

2.4.4 Value-based と Policy-based

MDP における最適戦略を求める強化学習の手法は、Q 学習だけではない。強化学習の手法は、DQN のような value-based な手法と、 policy-based な手法の 2 つに大別される。後者は方策勾配法 (policy gradient method) とも呼ばれる。

Q 学習のように価値関数を求め、間接的に最適戦略を求めようとする value-based な手法と異なり、方策勾配法では期待累積報酬を直接微分することで最適戦略を求めようとする。

戦略 $\pi(\theta)$ の期待累積報酬 $\rho(\pi)$ は、 $\rho(\pi) = V^\pi(s_0)$ と書ける。ここで、$V^\pi(s)$ は状態価値関数 (state-value function) であり、 $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$ を満たす。また、 $Q^\pi(s,a)$ は行動価値関数 (action-value function) であり、 $Q^\pi(s,a) = \mathbb{E}_{s' \sim T(s,a)} \left[R(s,a,s') + \gamma V^\pi(s') \right]$ を満たす。この式を θ で微分すると、いくつかの式展開の後に、以下の式が得られる [16]。

$$\frac{\partial \rho(\pi)}{\partial \theta} = \sum_s d^\pi(s) \sum_a \left[\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) \right] \tag{4.6}$$

ここで、$d^\pi(s)$ は割引率を加味した s での存在確率である。ここで、

$$\frac{\partial}{\partial \theta} \log \pi(s,a) = \frac{1}{\pi(s,a)} \frac{\partial \pi(s,a)}{\partial \theta} \tag{4.7}$$

であることを用いれば、上式は

$$\frac{\partial \rho(\pi)}{\partial \theta} = \sum_s d^\pi(s) \sum_a \left[\pi(s,a) \frac{\partial \log \pi(s,a)}{\partial \theta} Q^\pi(s,a) \right] = \mathbb{E}_s \mathbb{E}_a \left[\frac{\partial \log \pi(s,a)}{\partial \theta} Q^\pi(s,a) \right] \tag{4.8}$$

となる。この結果に従って勾配降下法で θ を学習させるのが方策勾配法である。

方策勾配法では、期待累積報酬を直接微分することで最適戦略を求めようとするため、Q 学習などと比べて学習途中の安定性が高いと言われている。また、Q 学習では扱えないような確率が 1 とならない戦略も扱うことができる。さらに、行動空間が連続になっても用いることができるという利点もある。

しかしながら、式 (4.8) を見るとわかるように、方策勾配法では学習データを学習対象の戦略 π に従って取らなければならない。すなわち、学習データの分布が学習対象の戦略に依存する。このように、学習対象の戦略に従って学習データをサンプルしなければならない手法は、 on-policy であると呼ぶ。これにより方策勾配法では ε-greedy などの探索手法は行えず、また過去の戦略で取得したデータもパラメータ更新後は用いることができないため、DQN などのように replay buffer を使うこともできない。さらに、データ間の相関を減らすためにゲームを並列に起動した場合、on-policy のためにパラメータの更新を同期させなければならず、パフォーマンスが落ちるという問題もある。逆に、 value-based な手法であれば、 on-policy でない手法もある。そのような手法は off-policy であるという。Q 学習は off-policy である。

なお、式 (4.8) の微分は分散が大きく、期待値をサンプルの平均で近似すると精度が大きく落ちてしまう。そのため、$Q^\pi(s,a)$ から 行動 a に依らない関数 $V^\pi(s)$ を引いた

$$\frac{\partial \rho(\pi)}{\partial \theta} = \mathbb{E}_s \mathbb{E}_a \left[\frac{\partial \log \pi(s,a)}{\partial \theta} \left(Q^\pi(s,a) - V^\pi(s) \right) \right] \tag{4.9}$$

を用いることが多い。この式も問題なく成り立つことが知られている。この $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ を advantage と呼ぶ。実際の方策勾配法では、データ $(s_t, a_t, r_t), \cdots, (s_{t+\tau}, a_{t+\tau}, r_{t+\tau})$ について、

$$A^\pi(s_t, a_t) = r_t + \gamma r_{t+1} + \cdots + \gamma^\tau V^\pi(s_{t+\tau}) - V^\pi(s_t) \tag{4.10}$$

とし、 V^π と π を同時に学習する手法が主流である。この手法を、π が実際に行動する actor 、V がそれを評価する critic という意味で、 actor-critic と呼ぶ。

2.4.5 Trust Region Policy Optimization と Proximal Policy Optimization

前項で述べたように、方策勾配法における on-policy の問題は非常に重く、様々な解決手法が提案されてきた。Schulman らが提案した Trust Region Policy Optimization (TRPO) [17] は、学習データの off-policy 性を概算し、学習対象の戦略を悪化させるような過度な摂動を伴うデータを棄却することで on-policy 性に対処する手法である。

具体的に、TRPO では以下の制約付き最大化問題を解く。

$$\begin{aligned} \underset{\theta}{\text{maximize}} \;\; & \mathbb{E}_{s,a\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{old}}(s,a)}A_{\theta_{old}}(s,a)\right] \\ \text{subject to} \;\; & \mathbb{E}_{s\sim\pi_{\theta_{old}}}\left[\mathrm{KL}\left(\pi_{\theta_{old}}(s,\cdot)||\pi_\theta(s,\cdot)\right)\right] \leq \delta \end{aligned} \tag{4.11}$$

ここで、$\mathrm{KL}(\cdot||\cdot)$ はカルバック・ライブラー情報量であり、θ_{old} は現在の学習対象の戦略のパラメータである。この式はつまり、現在のパラメータから一定距離以内であれば、直近の確率だけを補正することで学習を行うことができる、ということを表している。上式を制約がない状態で θ について微分すれば、$\pi_{\theta_{old}}$ による importance sampling を含めた式 (4.9) が得られる。

Schulman らはさらに、この手法をより簡便にした、 Proximal Policy Optimization (PPO) を提案した [18]。PPO では、計算コストの大きいカルバック・ライブラー情報量を単に確率の比のクリッピングに置き換えた。すなわち、PPO では式 (4.11) の代わりに、以下の目的関数を制約条件なしに最大化する。

$$L(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t\hat{A}_t,\, \mathrm{clip}\left(r_t,\, 1-\epsilon,\, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{4.12}$$

ここで、 $r_t = \frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_{\mathrm{old}}}(s_t,a_t)}$ は TRPO の式と同様に現在の確率の元の確率に対する比であり、 ϵ は比の上限を決めるしきい値、 $\hat{A}_t$ は advantage A_t の推定量である。この最適化問題は TRPO の最適化問題に比べて単純であり、かつ実験的に TRPO よりよい性能が出ることが知られている。

2.5 Counterfactual Regret Minimization

本研究では、不完全情報ゲームの解を強化学習を用いて求めることを目指している。不完全情報ゲームの分野においては情報集合の木を探索して解を求める Counterfactual Regret Minimization (CFR) [19, 20] と呼ばれる手法が主流であり、ゲームの状態遷移規則がわかる条件のもとで大きな

成果を上げている。また、本研究でも比較実験としてこの CFR をサンプリングで行えるようにした手法を実装し、比較している。そのため、本節ではこの CFR について簡単に説明する。

2.5.1 Regret Matching

展開型ゲームにおいて、時刻 T における regret R_i^T を

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} \left(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t) \right) \tag{5.1}$$

で定義する。この値は、相手の戦略を固定したときのプレイヤ i の期待報酬の最大値である。

二人零和ゲームにおいては、 regret が $R_i^T \leq \varepsilon$ を満たすとき、平均戦略

$$\bar{\sigma}_i^T(I)(a) = \frac{\sum_{t=1}^{T} p_i^{\sigma^t}(I)\, \sigma^t(I)(a)}{\sum_{t=1}^{T} p_i^{\sigma^t}(I)} \tag{5.2}$$

は 2ε-ナッシュ均衡となることが知られている。したがって、 regret を最小化するような戦略を計算することができれば、このゲームにおけるナッシュ均衡戦略を求めることができる。この理論を用いてナッシュ均衡戦略を求めることを、 regret matching と呼ぶ。先述した FP [8] も regret matching の一種である。

2.5.2 Counterfactual Regret Minimization

Lanctot らが提案した CounterFactual Regret Minimization (CFR) [19] は、展開型ゲームにおいて効率よく regret matching を行う手法である。

情報集合 I について、I から到達可能なすべての終端状態の集合を $Z_I = \{\, z \in Z \mid \exists h \sqsubseteq z, h \in I \,\}$ とし、その要素 $z \in Z_I$ について I に含まれる状態のうち z に到達可能なものを $z[I] \in I$ とする。情報集合はそのプレイヤから区別できない状態の集合であるが、完全記憶ゲームを仮定すればある終端状態に到達できる状態は高々ひとつしかない。

Counterfactual value $v_i(\sigma, I)$ を式

$$v_i(\sigma, I) = \sum_{z \in Z_I} p_{-i}^{\sigma}(z[I])\, p^{\sigma}(z \mid z[I])\, u_i(z) \tag{5.3}$$

で定義し、immediate counteractual regret $R_{i,imm}^T(I, a)$ を式

$$R_{i,imm}^T(I, a) = \frac{1}{T} \sum_{t=1}^{T} \left(v_i(\sigma_{(I \to a)}^t, I) - v_i(\sigma^t, I) \right) \tag{5.4}$$

で定義する。ここで、$\sigma_{(I_i \to a)}$ は、そのプレイヤ i がその情報集合 I においては行動 a を選択し、それ以外については戦略 σ に従って行動するような戦略プロファイルを表す。このとき、 regret と counterfacual regret の間に式

$$R_i^T \leq \sum_{I \in \mathcal{I}_i} R_{i,imm}^{T,+}(I) \tag{5.5}$$

が成立することが示せる。ここで、 x^+ は $\max(x, 0)$ を表す。

従って，ナッシュ均衡戦略を求めるためには、単に各情報集合に対して counterfactual regret $R_{i,imm}^{T,+}(I)$ を最小化するような戦略を計算すればよい。この戦略は情報集合の木を再帰的に辿ることで計算可能であり、これによって効率よくナッシュ均衡戦略を求めることができる。以上が CFR で用いられている手法である。

この手法を提案した Zinkevich らは、偶然手番における情報集合の遷移にサンプリングを用いたが、 Johanson らはこれを public chance と private chance に分類し、それぞれをベクトルで表現することで性能を向上させることに成功した [21]。その後、 Tammelin らが CFR を改良し提案した CFR+ [20] は、情報集合数が 10^{17} 程度ある 2 人リミットテキサス・ホールデムにおいて、十分小さな ε に対して ε-ナッシュ均衡解を求めることに成功した [22]。

アルゴリズム 2 Newral Fictitious Self-Play (NFSP)

Require:

ゲーム Γ，プレイヤ人数 $N \geq 2$

Ensure:

$\Pi\left(s, a|\theta^{\Pi}\right)$ は近似的なナッシュ均衡戦略を返すネットワーク

```
function NFSP(Γ)
    Initialize Π, Q, θ^Π, θ^Q, θ^Q', M^SL, M^RL
    for i = 1, 2, ···, N do                                  ▷ 各プレイヤについて
        Initialize M_i
    end for
    for iteration = 1, 2, ··· do
        ε ← ε(iteration)                                     ▷ ε-greedy に用いる ε
        for i = 1, 2, ···, N do                              ▷ 各プレイヤについて
            σ_i ← ε-greedy(Q) (確率 η) or Π (確率 1 − η)      ▷ 戦略を選択
        end for
        Initialize ゲーム Γ                                  ▷ ゲームを初期状態に戻す
        repeat                                               ▷ ゲームが終端するまで続ける
            n ← turn player of Γ
            M_n.s ← M_n.s'                                   ▷ 遷移前の状態を更新
            observe 状態 s* and M_n.s' ← s*                  ▷ 現在の状態を観測
            Store M_n in M^RL                                ▷ 学習データを保存
            if σ_n = ε-greedy(Q) then                        ▷ Q 学習の戦略に従ったデータのみ保存
                Store M_n in M^SL
            end if
            Periodically update θ^Q with M ∼ M^RL , θ^Q'     ▷ ネットワークを更新
            Periodically update θ^Π with M ∼ M^SL
            Periodically update θ^Q' ← θ^Q                   ▷ 一定回数の θ^Q 更新毎にターゲットを更新
            Sample アクション a from 戦略 σ_i                ▷ 戦略に従ってアクションを選択
            Execute アクション a on ゲーム Γ                 ▷ アクションを実行
            M_n.r ← 0                                        ▷ 終端状態以外では報酬は 0
        until Γ is over                                      ▷ ゲームが終端するまでループ
        for i = 1, 2, ···, N do                              ▷ 各プレイヤについて
            set 報酬 M_i.r                                   ▷ ゲームの終端状態に応じて報酬を設定
        end for
    end for
end function
```

第3章　NFSPの改良についての試行錯誤

3.1　NFSPと他手法の比較実験

本研究ではまず、一般の不完全情報ゲームを解くための手法としてNFSPが適切な手法であるかを確認するために、他の手法との比較実験を行った。

比較にあたって、本研究では環境の状態遷移規則を知ることなく学習を行う手法を検討している。そのため、CFRのようなゲームの木を探索する手法は用いることができない。これに対して、Monte Carlo CFR (MCCFR) [23] のようなサンプリングによってCFRを近似する手法であれば、環境の遷移を全てサンプリングで行うことによって、状態遷移規則を知ることなく戦略を求めることができる。また、強化学習を用いて単に自己対戦を行った場合でも、収束する保証はないものの、状態遷移規則を用いずに戦略の計算を行うことができる。

これらのことから、本節ではMCCFR、Q学習による自己対戦、FSP、NFSPの4手法について比較を行った。

3.1.1　実験の内容

実験では、不完全情報ゲームとしてKuhn poker [24] およびLeduc Hold'em [25] を用いた。Kuhn pokerは、後述する通り2人の小規模な不完全情報ゲームである。Leduc Hold'emは、public cardを1枚追加し、Kuhn pokerをより通常のHold'emに近づけた2人不完全情報ゲームである。これらのゲームの複雑さの指標として、情報集合数の一覧を表1.1に示す。Kuhn poker及びLeduc Hold'emは、Hold'emの重要な要素を残しつつ複雑性を減らしたゲームであると言える。

実験の対象としてこれらの単純化したゲームを用いたのは、可搾取量によって戦略を定量的に評価するためである。可搾取量は与えられた戦略プロファイルがどの程度ナッシュ均衡戦略から離れているかを定量的に示す正の値で、最適応答戦略に1ゲームあたりいくら搾取されるかを表している。可搾取量の計算にはゲームの木を探索する必要があるため、可搾取量の時間計算量はゲームの情報集合数に比例する。したがって、この値は大規模なゲームに対しては計算することができない。本論文では、可搾取量をグラフにプロットして戦略を定量的に評価するため、情報集合数の少ない単純化したゲームを用いた。

表 1.1: 各ゲームの情報集合数

Kuhn poker	Leduc Hold'em	HULHE	NLHE
10^1	10^2	10^{13}	10^{160}

表 1.2: 各ゲームにおけるランダム戦略の可搾取量

Kuhn poker	Leduc Hold'em
0.917	4.77

実験において、FSP の強化学習部分には Q 学習を、教師あり学習には行動の単純平均を用いた。Q 学習および FSP の強化学習部分では、学習率を 0.1、ε-greedy における ε を $\varepsilon = 0.1$ とした。これ以外の FSP および NFSP のハイパーパラメータは [26] と同じ設定にした。MCCFR のサンプリング戦略では $\varepsilon = 0.6$ とした。

3.1.2　実験の結果

Kuhn poker における実験の結果を図 1.1 に、Leduc Hold'em における結果を図 1.2 に示す。また、可搾取量の目安を示すため、各情報集合において合法手をランダムに選択する戦略プロファイルの可搾取量を表 1.2 に示す。

Q 学習の自己対戦について、Q 学習は相手の戦略に対する最適応答戦略を求めようとするため、理想的に学習が収束すればその戦略プロファイルはナッシュ均衡戦略となる。しかしながら、一般に不完全情報ゲームのナッシュ均衡戦略は純粋戦略であるとは限らない。すなわち、決定的な戦略の範囲内ではナッシュ均衡戦略は一般に存在するとは限らない。Q 学習は決定的な戦略しか求めることができないため、この手法は正しく収束しないと考えられる。また自己対戦自体も収束の保証はなく、実際に実験結果を見ると可搾取量は非常に高い値になっていることがわかる。FSP と MCCFR は、学習が進むにつれて可搾取量が減少しており、戦略がナッシュ均衡戦略に近づいていることがわかる。NFSP も学習が進むにつれて可搾取量が減少しているが、Kuhn poker では約 5×10^5 回、Leduc Hold'em では約 2×10^6 回の episode 付近で可搾取量の減少が止まっていることがわかる。これは、NFSP がニューラルネットワークによる関数近似を行っているため、関数近似を行っていない他の手法に比べて精度が落ちてしまったからではないかと考えられる。

また、この実験設定における各手法の実行時間を表 1.3 に示す。計測のために 10^6 episodes の実験を 10 回行い、その平均を取り、Q 学習の実行時間を 1 とする比率で示した。表 1.3 から分かる通り、NFSP は他の手法に比べて非常に低速であることがわかる。これは、ニューラルネットワークを用いるために学習の各ステップで大規模な行列計算が入ってしまうことが原因であると考えられる。

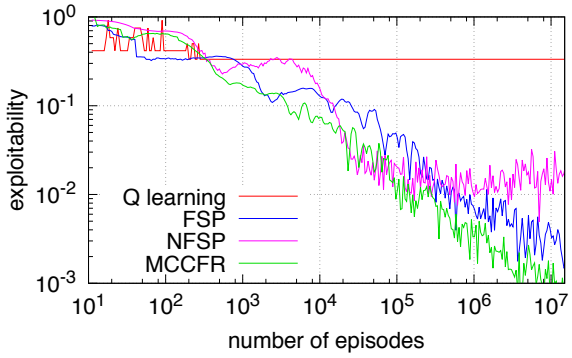

図 1.1: Kuhn poker における可搾取量

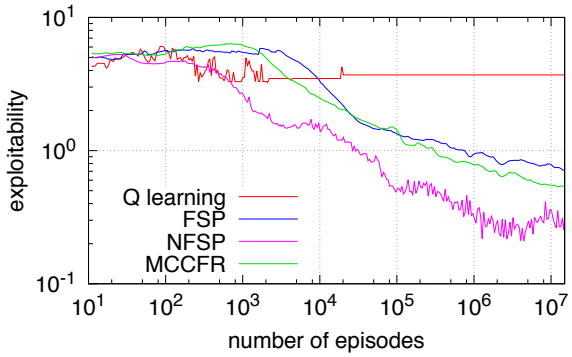

図 1.2: Leduc Hold'em における可搾取量

以上のことから、ニューラルネットワークによる近似を含まない FSP は、CFR を元にしたアルゴリズムである MCCFR と比較しても遜色ない戦略を獲得できるにもかかわらず、FSP にニューラルネットワークを加えた NFSP は可搾取量の減少が途中で止まってしまうことがわかった。このため、本研究では CFR とニューラルネットワークを組み合わせた新たな手法を研究するのではなく、FSP を元にした NFSP に改良を加えることを目指した。

3.2　NFSP の平均戦略の計算の補正

本研究では次に、NFSP の平均戦略を計算する際の分布のずれに着目し、その補正を試みた。

表 1.3: 各手法の実行時間

	Kuhn poker	Leduc Hold'em
Q 学習	1.0	1.0
FSP	2.0	2.2
NFSP	5.3×10^3	4.5×10^3
MCCFR	2.0	2.0

3.2.1 問題点

FSP では、FP における平均戦略の計算を展開型ゲームでも行うために、各情報集合での平均を計算する際にその情報集合への到達確率を掛けることで補正を行っている。具体的には、標準型ゲームにおける任意の 2 つの混合戦略 $\pi_i^{N,1}, \pi_i^{N,2}$ と、対応する展開型ゲームにおけるその混合戦略と実現等価な 2 つの戦略 π_i^1, π_i^2 について、混合戦略 $\pi_i^N = \alpha\pi_i^{N,1} + (1-\alpha)\,\pi_i^{N,2}$ と実現等価な戦略 π_i を

$$\forall u \in \mathcal{U}, a \in \mathcal{A}, \pi_i(u,a) \propto \alpha x_i^{\pi_i^1}(u)\pi_i^1(u,a) + (1-\alpha)\, x_i^{\pi_i^2}(u)\pi_i^2(u,a) \tag{2.1}$$

と構成することができることを用いている。ここで、$x_i^\sigma(u)$ はプレイヤ i が戦略 σ に従った時に情報集合 u にたどり着くために必要な行動を全て選択する確率である。完全記憶ゲームを仮定すれば、プレイヤ i から見た情報集合 u にたどり着くために必要な情報集合と行動の列はプレイヤ i にとって一意に定まるため、その確率を全て掛けた値が $x_i^\sigma(u)$ となる。

ところが、NFSP の教師あり学習において、reservoir replay buffer が保持しているデータからサンプリングしたデータの分布はこの式 (2.1) を満たさない。いま、戦略プロファイル π に従って 1 エピソードのゲームを行ったとき、情報集合 u において行動 a を選択するデータ (u,a) が得られる確率を考えれば、その確率は $x_i^\pi(u)x_{-i}^\pi(u)\pi_i(u,a)$ となる。式 (2.1) と見比べれば、$x_{-i}^\pi(u)$ が一定であればこの分布は式 (2.1) と等しくなることがわかる。しかし、実際には学習中に $x_{-i}^\pi(u)$ は変化するため、この分布は期待する分布とは一致しない。

3.2.2 試した手法

この分布のずれを補正するために、reservoir replay buffer にデータを追加する際の確率を変更することで補正を試みた。

具体的には、reservoir replay buffer にデータ $d = (I_i(s), a)$ を追加する際に、priority $p(d) = \frac{1}{x_{-i}^\pi(s)}$ を与え、確率 $\frac{C \cdot p(d)}{\sum_k p(d_k) + p(d)}$ でデータを buffer に追加する。ただし、C は buffer のサイズであり、$\sum_k p(d_k)$ は棄却したデータも含む全てのデータについての和を取る。このようにすることで、$\sum_k p(d_k) + p(d) > C \cdot p(d)$ が成り立つ限り、データ d が buffer に入っている確率は $p(d)$ に比例す

る。なぜなら、データ d_{t+1} がまず格納される確率は

$$\frac{C \cdot p(d_{t+1})}{\sum_{\tau=1}^{t} p(d_\tau) + p(d_{t+1})} \tag{2.2}$$

であり、次のデータ d_{t+2} に d_{t+1} が押し出されない（d_{t+1} が buffer に残る）確率は

$$1 - \frac{1}{C}\frac{C \cdot p(d_{t+2})}{\sum_{\tau=1}^{t+1} p(d_\tau) + p(d_{t+2})} = \frac{\sum_{\tau=1}^{t} p(d_\tau) + p(d_{t+1})}{\sum_{\tau=1}^{t} p(d_\tau) + p(d_{t+1}) + p(d_{t+2})} \tag{2.3}$$

であるから、同様に考えてデータ d_{t+1} が buffer にある確率は常に $\frac{C \cdot p(d_{t+1})}{\sum_d p(d)}$ となるからである。

この計算では、確率が 1 を超える場合を考慮していない。この確率が 1 を超えるということは、$x^{\pi}_{-i}(s)$ が他のデータに比べて非常に小さいということを表している。このようなデータの対処方法についてはいくつか手法が考えられるが、今回は単に確率の上限を 1 として計算を行った。

また、この実験に加えて、教師あり学習を純粋な平均に置き換える実験も行った。この実験では、平均戦略の計算を各情報集合に対するテーブルを保持することで行った。こうすることで、教師あり学習が理想的に平均を計算できた場合の結果を見ることができ、より詳細な考察が行えるようになると考えられる。また、この純粋な平均の元で先述の補正を加えることで、教師あり学習で起こっていたようなデータのサンプル効率の低さや確率の上限の問題などが解消され、この確率の補正が結果にどの程度影響を与えているかがよりわかりやすくなると考えられる。

3.2.3　実験の内容

前節と同様、Leduc Hold'em を用いて戦略の可搾取量を計測した。ハイパーパラメータについては、基本的には前項と同様 [26] のハイパーパラメータを使用した。ただし、教師あり学習の最適化手法を Adam [27] ではなく学習率 0.005 の SGD (Stochastic Gradient Descent) にしている。

3.2.4　実験の結果

実験の結果を図 2.1 に示す。通常の NFSP の結果が赤線、補正を行った結果が緑線、平均戦略を単純平均で計算した結果が黄線、単純平均に補正を行った結果が茶線で示されている。

まず通常の NFSP と補正を加えたものを比較すると、補正を加えた手法のほうが結果が悪くなっていることがわかる。また、平均戦略を単純平均で計算した手法を比較しても、補正を加えた手法のほうが結果が悪くなっている。単純平均に補正を加えた手法では、3×10^5 ゲーム付近と 1×10^7 ゲーム付近で可搾取量が増加する現象が見られた。

補正を加えたほうが結果が悪化してしまったことについて、平均を計算するデータの分散が大きくなってしまったことが原因の一つとして考えられる。到達確率を補正することで、データの期待値

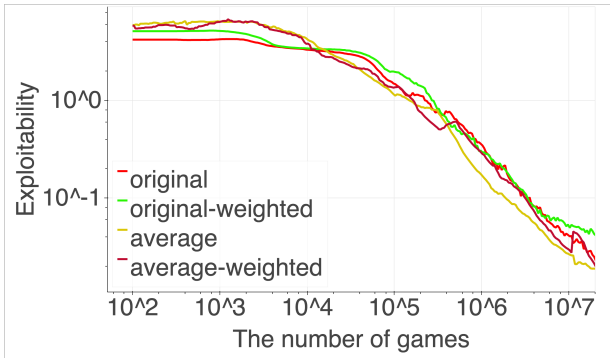

図 2.1: 平均戦略の計算を補正した NFSP の可搾取量

は真の戦略の平均と一致するようになったが、相手の到達確率 $x^{\pi}_{-i}(s)$ が小さい状態のデータには非常に大きな値が掛かるため、データの分散が大きくなり、収束が遅くなってしまったのではないかと考えられる。

また、 reservoir replay buffer の採択率に補正を加えた手法では、採択率の上限を 1 としたため前述のような問題は起こらない。しかし、到達確率の高いゲーム開始直後の状態などが相対的に棄却されやすくなるために、教師あり学習が新しい最適応答戦略を平均するのに時間がかかってしまい、結局学習が遅くなってしまうのではないかと考えられる。

3.3　探索由来のデータを含まない教師あり学習

次に、NFSP の教師あり学習が最適応答戦略の平均戦略を計算することを意図していることに着目し、教師あり学習の学習データに探索で得たデータを含めないようにすることを考えた。

3.3.1　問題点

NFSP の強化学習では、相手プレイヤの戦略に対する最適応答戦略を求めるために、探索と活用を行っている。具体的には、NFSP では ε-greedy を用いている。すなわち、確率 ε でランダムな手を選択し、確率 $1-\varepsilon$ で Q 値が最大であるような手を確定的に選択する。

ε-greedy を用いた Q 学習では、学習中は探索を含めた方策に基いて行動する。しかし、Q 学習は off-policy な学習方式で、Q 値が示す値は行動方策ではなく greedy に行動を選択した場合の期待累積報酬なので、最適応答戦略としては探索を含まない greedy な戦略を用いるべきである。探索を含めた方策を最適応答戦略として用い、平均戦略の計算に含めてしまうと、これがノイズとなって戦略がナッシュ均衡から外れてしまうことが考えられる。

オリジナルの NFSP では、ε-greedy の ε を学習の進行に合わせて小さくしているため、学習が進むにつれて探索を含めた戦略が greedy な戦略に近づいていき、先述の問題は起こらない。しかしながら、NFSP で行われる自己対戦は対戦相手の戦略が学習の進行に応じて変化するため、通常の Q 学習と異なり探索を無くすことはできないと考えられる。例えばある時刻 t について、状態 s で行動 a_1 を取る価値が a_2 よりも低いとすると、強化学習が正しく機能していれば平均戦略の a_1 を選択する確率は 0 に近づいていく。ここで、対戦相手の戦略が変化し、a_2 の価値は変化しないまま a_1 の価値が a_2 よりも高くなった場合、そのことを学習するためには s において a_1 を選択しなければならない。しかし、ε が十分小さくなってしまった場合、もはや a_1 が選択されることはなく、最適応答戦略を正しく求めることができなくなってしまう。

3.3.2　試した手法

これらの問題を同時に解決するためには、探索の確率 ε をある程度大きな値に保ちつつ、活用によって得られたデータのみを教師あり学習に与えればよい。もし探索を含めた方策が最適応答戦略になっているのであれば、探索によって得られたデータを教師あり学習に含めないことで、学習が遅くなってしまう。しかし、ε-greedy を用いた Q 学習では、探索を含めた方策は一般に最適方策にはなっていないので、このようなことは起こらないと考えられる。

以上を踏まえた提案手法の擬似コードを Algorithm 3 に示す。提案手法に関わる部分は太字で示している。

3.3.3　実験の内容

前項と同様、 Leduc Hold'em を用いて戦略の可搾取量を計測した。

本研究では、教師あり学習の学習データに強化学習の探索で得たデータを含めないようにすることで、性能を悪化させることなく継続的に探索が行えるようになる、という仮説を立てている。そのため、比較する実験設定として以下の 4 つを用意した。

A. 通常の NFSP。ε は学習とともに急激に減衰する。具体的には、$\varepsilon = \frac{0.06}{\sqrt{n}}$、$n$ は [3] における iteration (128 game steps)。

B. A. において、教師あり学習の学習データに強化学習の探索で得たデータを含めないようにしたもの（提案手法)。

C. A. において、ε を減衰させず、固定したもの。

D. B. および C. 、すなわち、教師あり学習の学習データに強化学習の探索で得たデータを含めないようにし、さらに ε を固定したもの。

表 3.1: 各設定における、10^7 ゲーム行った後の可搾取量の値。各設定につき 5 回ずつ実験を行い、その平均を記載している。通常の NFSP (A.) と比較して、有意水準 $\alpha = 0.01$ で有意に差があるものには † を付けている。

設定	可搾取量
A.	$(3.8 \pm 0.5) \times 10^{-2}$
B.	$(4 \pm 1) \times 10^{-2}$
C.	$(26.8 \pm 0.8) \times 10^{-2\dagger}$
D.	$(\mathbf{2.3} \pm \mathbf{0.5}) \times \mathbf{10^{-2}}^{\dagger}$

ε の固定値には 0.1 を用いた。それ以外のハイパーパラメータは [3] に従っている。

3.3.4 実験の結果

それぞれの設定について、10^7 ゲーム行った後の可搾取量を 5 回計測した平均値を表 3.1 に示す。ε が減衰する設定では通常の NFSP と提案手法に差はないが、ε を固定値にした場合、通常の NFSP では性能が大きく悪化するが、提案手法では性能が有意に改善することがわかる。これは、通常の NFSP では探索の結果も平均に含まれるため、一定確率で探索を行うような設定では平均戦略にノイズが入ってしまうが、提案手法ではそのようなことが起こらず、学習が進んでも探索を続けられるからであると考えられる。

また、性能に対する ε の影響を調べるため、$0.01, 0.02, 0.05, 0.1, 0.2, 0.5$ の 6 種類の ε に対して同様の実験を行った。その結果を図 3.1 に示す。

既存の設定では、ε を固定すると性能が大きく悪化し、その度合いが ε の増加に従うことがわかる。これは、既存の設定では探索の結果も平均戦略に含まれてしまうため、探索を行いすぎると性能が悪化する、という直感に即している。また、提案手法では、ε を固定すると性能が多少改善し、ε が 0.02 から 0.1 の間にある場合に特に性能に改善が見られるという大まかな傾向が見て取れる。これは、平均戦略の学習に探索を含めないため、探索を続けることで平均戦略の性能を落とすことなく最適応答戦略を正しく計算できるのではないかという直感に即している。ε が大きすぎる場合に性能が多少悪化するのは、活用で得られるデータが減少することで教師あり学習に学習させるデータの量が減ってしまい、教師あり学習の追従速度が低下するためであると考えられる。

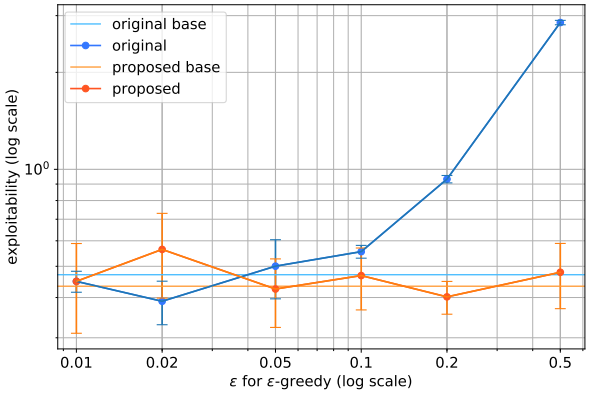

図 3.1: 各設定における、10^7 ゲーム行った後の可搾取量の値。名前に base が付く横線は、オリジナルの NFSP と同様、ε が急速に減衰する設定における結果を表している。 $\varepsilon = 0.1$ の設定と ε が減衰する設定については 5 回ずつ実験を行った平均とその標準偏差を、それ以外のデータについては 1 回だけ実験を行った結果を表示している。

アルゴリズム 3 探索データを含まない教師あり学習を用いた NFSP

```
Γ is a game and N is the number of players
function NFSP(Γ)
    Initialize Π, Q、θ^Π, θ^Q, θ^Q', M^SL、M^RL
    for i = 1, 2, ···, N do                                  ▷ 各プレイヤについて
        Initialize M_i
    end for
    for iteration = 1, 2, ··· do
        ε ← ε(iteration)                                     ▷ ε の値
        for i = 1, 2, ···, N do
            σ_i ← ε-greedy(Q) (確率 η) or Π (確率 1 − η)
        end for
        Initialize Γ
        repeat
            n ← turn player of Γ
            M_n.s ← M_n.s'
            observe state s* and M_n.s' ← s*
            Store M_n in M^RL                                ▷ 無条件でデータを保存
            if M_n.IsGreedy then        ▷ greedy に行動している強化学習由来のデータのみ保存
                Store M_n in M^SL
            end if
            Periodically update θ^Q with M ∼ M^RL , θ^Q'
            Periodically update θ^Π with M ∼ M^SL
            Periodically update θ^Q' ← θ^Q                   ▷ RL のターゲットを更新
            Sample action a by strategy σ_i
            Execute a on Γ
            M_n.a ← a
            M_n.IsGreedy ← IsGreedy        ▷ 強化学習でかつ greedy に行動している場合のみ
チェック
            Reward M_n.r ← 0                                 ▷ 終端状態以外では報酬 0
        until Γ is over
        for i = 1, 2, ···, N do
            set M_i.r                                        ▷ 終端では状態に従って報酬を与える
        end for
    end for
    return Π(s, a|θ^Π)
end function
```

第4章　NFSPを用いたRTSゲームの均衡解の計算

前章までで、本研究では NFSP のいくつかの問題点に着目し、それらを解決するための手法を考えて実装し、比較実験を行った。

本章では、この NFSP を用いて Real-Time Strategy (RTS) ゲームの均衡解を実際に計算し、得られた均衡解について定性的・定量的な評価を行う。また、強化学習が実際には off-policy ではないという点に注目し、方策勾配法と組み合わせた新しい NFSP を提案し、RTS ゲームに適用する。

4.1　RTS ゲーム

4.1.1　RTS ゲームの意義

近年の深層学習の発展によって、強化学習は多くの複雑なゲーム環境で成功を収めている。特に、Atari 2600 のゲーム環境では、強化学習によって画像情報のみからほとんどすべてのゲームについて人間を超えるスコアが達成されている [1, 28, 29]。しかしながら、リアルタイムストラテジー (RTS) ゲームの分野では、強化学習を用いた AI はまだ人間のトッププレイヤの実力を上回っていない [30]。RTS ゲームは、チェスや囲碁に続く多人数強化学習における次世代の挑戦対象であると考えられており [31, 32]、この RTS ゲームにおいて強いプレイヤを求める手法が望まれている。

大規模な RTS ゲームにおける研究を効率よく行うために、いくつかの研究用ゲーム環境が提案されている [33, 34, 30]。強化学習の研究を高速に行うためのプラットフォームの一つである ELF (Extensive, Lightweight, and Flexible platform) [31] では、 Mini-RTS と呼ばれる研究用の RTS ゲームが提供されている。このゲームは RTS ゲームとしては小規模であるものの、 fog-of-war やリソース管理など RTS ゲームの特徴をすべて保持しており、かつ既存の他のゲーム環境と比較して高速に動作する。本章では、より大規模な RTS ゲームを解くための第一歩として、この Mini-RTS におけるゲーム理論的な意味での解を NFSP を用いて求めることを目指す。

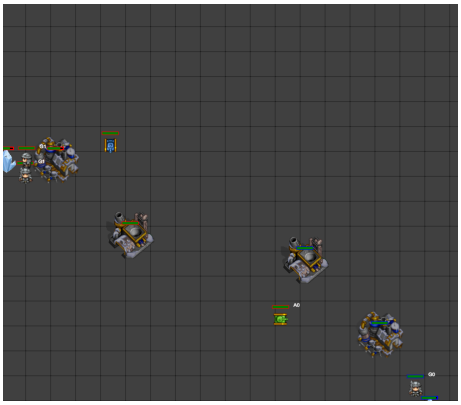

図 1.1: Mini-RTS の一場面

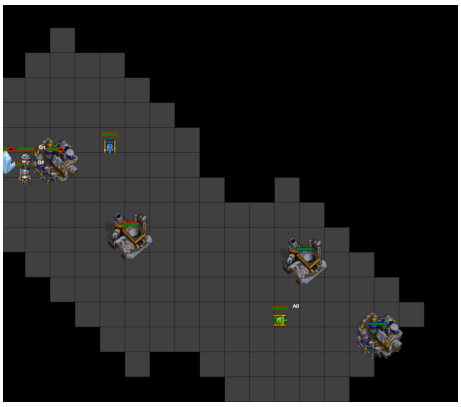

図 1.2: 同場面における fog-of-war

4.1.2 ELF Mini-RTS

本章の目的は、 ELF Mini-RTS において、任意のプレイヤに搾取されないような均衡戦略を求めることである。

Mini-RTS におけるプレイヤの目的は、戦車を作成し、相手の基地を破壊することである。各プレイヤは自分の基地、数種類のユニット、そしてリソースを持っている。プレイヤはリソースを消費することで兵士を作成でき、空き地に兵士を送りリソースを消費することで兵舎を作成でき、兵舎に兵士を送りリソースを消費することで戦車を作ることができる。

ゲーム画面の例を図 1.1 に示す。

Mini-RTS はフレーム単位で動作する。プレイヤは各フレームで各ユニットにどの行動を行うかの命令を送り、それらの行動が実行されて環境が遷移し、新たな観測が返る、というサイクルを繰り返すことでゲームが進行する。ゲームには fog-of-war があり、観測は自分のユニットがいる周囲についてのみ観測可能であり、他の部分はマスクが掛けられている。したがって、このゲームは不完全情報ゲームである。Fog-of-war の例を図 1.2 に示す。

「兵士 1 を 2 ピクセル右に移動させる」といった局所的な命令を指定する代わりに、 Mini-RTS では「空いている兵士に兵舎を建設させる」「基地を防衛する」といった大域的な命令を選択することができる。これらの命令は全ユニットに同時に作用する。簡単のため、本研究ではこれらの大域的な命令のみを用いた。具体的には、プレイヤは各フレームで 9 つの行動から 1 つを選択する。その内訳は、建設に関する行動が 4 種類（兵士・兵舎・近接戦車・遠隔戦車)、攻撃に関する行動が 4 種類（攻撃、離れて攻撃、攻撃して戻る、基地を防衛する)、何もしない行動が 1 種類となっている。

また、環境からプレイヤに与えられる観測は、ゲーム画面を 20×20 のグリッドに分割し、その各グリッドにあるオブジェクトや基地の残り耐久力などを表す $22 \times 20 \times 20$ の値である。

4.1.3 RTS ゲームの多人数性と関連研究

RTS ゲームにおける強いプレイヤを作成するためには、戦略と戦術のトレードオフ、リアルタイムな意思決定、事前知識の活用など、様々な難点が存在している [35, 36] が、本研究ではそのうち多人数性を主な研究対象とする。 RTS ゲームは二人以上の多人数ゲームであり、学習するプレイヤにとって環境が時間的に定常ではない。これは、強化学習が環境に対して設ける仮定の一つである Markov Decision Process (MDP) 性に反する。

このような環境に対処するために、 opponent modeling と呼ばれる手法が提案されている [37, 38]。この手法では、各プレイヤは対戦相手の戦略を推定し、その推定した戦略に対する自身の累積期待報酬を最大化するように学習する。対戦相手を推定した戦略に固定することで、環境を定常な MDP とみなして学習できるようになり、通常の強化学習の枠組みで議論することができるようになる。しかしながら、この手法では推定した戦略への最適応答戦略を求めることになってしまう。これはゲームの解を求めるという方針とは大きく異なっている。推定した戦略に対して特化することになるため、その学習した戦略を搾取するような戦略が存在する可能性が存在するからである。

また、多人数ゲームで強化学習を行う別の手法として、自己対戦 (self-play) が挙げられる [39, 32]。最も単純な自己対戦では、各プレイヤが同時に相手プレイヤに対する報酬を最大化するように学習する。自己対戦では、各プレイヤが合理的に (rational) 学習していて、かつその学習が収束すれば、その戦略の組はナッシュ均衡になることが知られている [40]。しかしながら、自己対戦では学習が収束する保証はなく、実際に収束せず振動する例も観測されている [41]。この自己対戦による学習を安定させるために、様々な手法が提案されている。 Foerster らは、学習で用いられる experience replay に importance sampling を用いることで、自己対戦における各プレイヤから見た時間的非定常性を補正する手法を提案した [42]。 また、 Foerster らは、自己対戦の相手も自分と同じように学習しているという事実を用い、対戦相手の学習の勾配を自身の学習に用いることで学習を安定化させる手法を提案した [43]。

Heinrich らは、これらの手法とは根本的に異なるアプローチとして、ゲーム理論の Fictitious Play (FP) に相当するアルゴリズムを機械学習を用いて行う Fictitious Self-Play (FSP) と呼ばれる手法を提案した [10]。FSP では、各プレイヤは強化学習を用いて対戦相手の平均戦略に対する累積期待報酬を最大化するような学習を行い、同時に教師あり学習を用いてその強化学習で得られた戦略を平均するような戦略を学習する。これは FP で行う最適応答戦略の計算と平均戦略の計算を近似するようになっており、かつ機械学習によって状態のテーブルを持てないようなゲームにも適用できるようになっている。FP はいくつかの条件の元でナッシュ均衡戦略に収束することが証明されているため、その近似である FSP も通常の自己対戦と比べてより収束する可能性が高いと考えられる。実際に、 FSP に深層学習の手法といくつかの改良を加えた NFSP (Neural Fictitious Self-Play) は、単純なポーカーのゲーム環境において、事前知識を用いることなく近似的なナッシュ均衡戦略を得ることに成功した [3]。

本研究では、この NFSP に対して方策勾配法 (policy gradient method) が適用でき、かつその手法を用いて Mini-RTS の均衡解を得ることができることを示す。また、方策勾配法と組み合わせることによって、 NFSP のネットワークを通常の self-play の学習で事前学習できるようになり、それによって NFSP の学習速度を向上させられることを示す。この研究は、我々の知る限り初めて非自明な RTS ゲームの均衡解を求めることに成功した研究であり、したがってより複雑な RTS ゲームを解くための重要な第一歩である。

4.2 提案手法

4.2.1 Off-policy 性が使えないことに着目した方策勾配法の適用

Heinrich らは、NFSP の強化学習の手法として DQN を用いた [3]。DQN は value-based な手法であり、従って本来は off-policy である。NFSP ではこの性質を利用して、通常の DQN と同じように circular replay buffer を用いて過去の履歴を保存し、サンプル効率を高めていた。

しかしながら、NFSP の学習では、強化学習の対戦相手の戦略が変化する。なぜなら、NFSP の強化学習は相手の平均戦略に対する最適応答戦略を近似するように学習を行うため、学習の際の環境が対戦相手の教師あり学習の戦略に依存するからである。従って、NFSP における DQN の circular replay buffer の使用は本来は適切ではない。

DQN のような value-based な手法は、policy-based な手法と比べて累積報酬を直接最大化しているわけではないという欠点がある。その代わりに、policy-based な手法とは異なり現在の方策に従ってデータをサンプルしなくてもよいため、circular buffer のように過去の方策でサンプルしたデータもそのまま学習に用いることができるという利点がある。

しかし、前述したように、今回の設定では過去のデータをそのまま学習に用いるのは適切ではない。以前のデータを学習に用いることは、相手の戦略が時間的に定常であることを仮定することになるが、実際には相手の戦略は（特に、DQN のような greedy な戦略では）大きく変化しうるためである。したがって、value-based な手法の利点は活用できないことになる。このために、本章では NFSP の強化学習の手法として、DQN ではなく方策勾配法の一種である PPO を用いた。

また、NFSP の強化学習に方策勾配法を用いることで、別の利点も挙げられる。完全情報ゲームでは、ナッシュ均衡であるような戦略プロファイルの中に、すべての状態において確定的に行動を決定するような戦略プロファイルがあることが知られている。しかしながら、不完全情報ゲームにおいては、そのような戦略プロファイルは一般には存在しない。したがって、例えば DQN のような確定的な戦略しか取ることのできない強化学習の手法では、純粋な自己対戦によってナッシュ均衡を得ることは難しいと考えられる。実際、第 3.1 節で行った実験では、DQN の自己対戦で得られた戦略プロファイルの可搾取量は非常に高かった。一方、方策勾配法は確率的な戦略も表現することができる。もしニューラルネットワークの表現力が十分に高ければ、方策勾配法は任意の確率的な方

策の中から環境における最適方策を計算するので、不完全情報ゲームであっても通常の自己対戦でナッシュ均衡戦略を得られる可能性がある。したがって、NFSP だけでなく、純粋な自己対戦もこの RTS ゲームに適用することができ、NFSP によって得られた戦略と比較を行うことができる。

4.2.2 NFSP の RTS ゲームへの適用

NFSP を RTS ゲームに適用し、その結果を観測することが本章の目的である。NFSP は二人零和完全記憶展開型ゲームに対して適用できる手法である。したがって、この手法を RTS ゲームに適用するにあたって、RTS ゲームが二人零和完全記憶展開型ゲームであることを確認する必要がある。

RTS ゲームには多くのユニットが存在しているが、プレイヤはすべてのユニットの観測を得ることができ、かつすべてのユニットに命令を出すことができる。したがって、プレイヤが二人であれば、勝敗のみを報酬とすることでこのゲームは二人零和ゲームとみなすことができる。また、先述した通り、 fog-of-war によって観測が制限されているため、このゲームは不完全情報ゲームである。ほとんどの RTS ゲームは離散的なフレームで管理されており、プレイヤの意思決定が十分速ければこれは順番に手番を行うゲームであるとみなせるため、このゲームは展開型ゲームであるとしてもよい。

ゲームが完全記憶であるかどうかは、プレイヤが過去の観測をすべて記憶しているかどうかに依存する。プレイヤが過去の観測を再帰構造などで記憶している場合、環境から得られる観測が過去の情報を含んでいなかったとしても、このゲームを完全記憶ゲームであるとみなすことができる。例えば Deep Reccurent Q-Network (DRQN) [44] などはその例である。DRQN では、意思決定を行う際に recurrent neural network を用いることで、部分観測な 1 人ゲーム（POMDP とみなせる）を効率よく解くことができる。

しかしながら、NFSP と再帰構造を組み合わせることは容易ではない。再帰構造を含めて観測を表現する場合、ゲーム中のある観測を記憶するためにはそれ以前のすべての観測を記憶する必要がある。すなわち、例えばあるプレイヤにとって観測の列 $o_1, o_2, \cdots, o_t$ があったとき、このゲームを完全記憶ゲームにするためにはすべての o_i を用いて o_t に相当する状態表現を得る必要がある。しかし、NFSP では教師あり学習のために reservoir replay buffer を用いるため、この o_t に相当する観測列を replay buffer に保存する必要がある。この状態表現は学習中に変化しうるため、replay buffer にはすべての o_i を保存しなければならない。したがって、再帰構造を NFSP と組み合わせる場合、教師あり学習の replay buffer のサイズがゲームの深さに反比例してしまい、スケーラビリティが失われてしまう。DRQN においても、replay buffer を用いる関係上同様の問題は存在している。しかし、DQN における reolay buffer の役割はあくまでも観測間の相関を減らすことであり、replay buffer のサイズが小さくなっても本質的に手法に問題が生じるわけではない。これに対して NFSP では、教師あり学習が過去すべての手を模倣するように学習しなければならないため、replay buffer のサイ

ズが小さくなることで性能は大きく下がると考えられる。以上のことから、NFSP に対して DRQN のようなアプローチで完全記憶性を担保することはできない。

本研究では、再帰構造などを用いることなく、単に環境から得られる観測のみで意思決定を行った。したがってこのゲームは完全記憶ゲームではなく、本質的には NFSP が正しくナッシュ均衡戦略を求められる保証はない。しかしながら、強化学習のベンチマークとして用いられる多くの環境が本質的には部分観測ゲームであるにもかかわらず、それらのベンチマークにおいて MDP であるという仮定を用いた強化学習の手法が良い成績を残していることを考えると、部分観測性は実験的にはあまり影響を及ぼさないのではないかと考えることもできる。本論文では完全記憶性についてはこれ以上の議論は行わない。

4.2.3 アルゴリズム

本章で用いる、 NFSP と PPO を組み合わせたアルゴリズムをアルゴリズム 4 に示す。

このアルゴリズムでは、行動を決定する関数と学習を行う関数がコールバックとして各 Γ_i に登録される。10 行目の各ステップにおいて、各 Γ_i では複数のゲームスレッドが同時に実行され、いずれかのコールバックが適切なバッチとともに呼び出される。

本研究では、 ゲームインスタンス Γ_i を各プレイヤ p について並列に作成し、実行した。各インスタンスではプレイヤ p が強化学習に、それ以外のプレイヤが教師あり学習に従って行動し、各 Γ_p ではプレイヤ p のみが学習を行うようにした。このアルゴリズムは、各ゲームの先頭で強化学習か教師あり学習を選択し、各プレイヤを等価に扱うという元の NFSP のアルゴリズムとは大きく異なっている。元のアルゴリズムでは、各プレイヤ p は 4 種類の経験データを得ることができる。すなわち、 (π_p, π_{-p}) 、 (π_p, β_{-p}) 、 (β_p, π_{-p}) 、 (β_p, β_{-p}) である。ここで、 π は教師あり学習を、 β は強化学習を表す。また、 $-p$ はプレイヤ p を除く全プレイヤを表す。もし強化学習が off-policy であれば、自身が β_p で行動したデータだけでなく、 π_p に従って行動したデータも強化学習に用いることができる。しかしながら、もし強化学習が on-policy であれば、 $(\pi_p, \cdot_{-p})$ は強化学習の学習には用いることができなくなってしまう。教師あり学習も自身が π_p に従って行動したデータは学習に用いないので、強化学習が on-policy の場合、元のアルゴリズムのままでは多くのデータが無駄になってしまう。これを避けるため、アルゴリズム 4 では学習するプレイヤのみが強化学習に従って行動を行うようになっている。

16 行目において、 $\mathcal{L}_{RL}$ は PPO と同様に、以下の式で計算される。

$$\mathcal{L}_{RL} = \mathcal{L}_{policy} + \alpha\mathcal{L}_{entropy} + \beta\mathcal{L}_{value} \tag{2.1}$$

アルゴリズム 4 NFSP with PPO

$\Gamma = \{\Gamma_i \mid i = 1, 2, \cdots, N\}$ is a set of Game instances and N is the number of players
function MAIN(Γ, N)
　for $p = 1, 2, \cdots, N$ in parallel **do**　　▷ p is a learning player
　　Γ_p.Initialize()
　　Γ_p.RegisterCallback(p, Trainer)
　　Γ_p.RegisterCallback(p, RLActor)
　　for $q = 1, \cdots, p-1, p+1, \cdots, N$ **do**
　　Γ_p.RegisterCallback(q, SLActor)
　　end for
　　repeat
　　Γ_p.DoStep()　▷ Multiple games are executed concurrently and a corresponding callback is called at a step
　　until Time steps exceed the certain limits
　end for
end function
function TRAINER(p, batch)
　$\{S_\tau, A_\tau, \Pi_\tau, R_\tau\}_{\{\tau=t,\cdots,t+T-1\}} \leftarrow$ batch
　　▷ State, action, probability distribution, and reward
　　▷ Π_t is a probability distribution of NN_{RL} at t
　Calculate $\mathcal{L}_{RL}$ with eq. (2.1)
　Memorize $\{S_\tau, \Pi_\tau\}$ in buffer $\mathcal{M}_{SL}$
　Sample $S, \Pi \leftarrow \mathcal{M}_{SL}$
　Calculate $\mathcal{L}_{SL}$ with closs entropy
　Optimize NN_{RL} and NN_{SL} with $\mathcal{L}_{RL}$ and $\mathcal{L}_{SL}$
end function
function RLACTOR(p, batch)
　$S \leftarrow$ batch
　$\pi \leftarrow \mathrm{NN}_{RL}(S)$
　return Sampled $a \leftarrow \pi$
end function
function SLACTOR(p, batch)
　$S \leftarrow$ batch
　$\pi \leftarrow \mathrm{NN}_{SL}(S)$
　return Sampled $a \leftarrow \pi$
end function

ここで、 $\mathcal{L}_{policy}$ は第 2.4.5 項で述べた PPO の目的関数

$$L(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t \hat{A}_t,\, \mathrm{clip}\left(r_t,\, 1-\epsilon,\, 1+\epsilon\right) \hat{A}_t \right) \right] \tag{2.2}$$

を -1 倍したものであり、 $\mathcal{L}_{entropy} = \sum_a \pi_\theta(a \mid s) \log \pi_\theta(a \mid s)$ はプレイヤに探索を行わせるための正則化項である。また、 $\mathcal{L}_{value}$ は V_θ と V_{target} のクリッピング付き二乗誤差である。具体的には、式 (2.2) において、推定量 $\hat{A}$ は

$$\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$$

$$A_t = \delta_t + k\delta_{t+1} + \cdots + k^{T-t+1}\delta_{T-1}$$

$$\bar{A}_t = \frac{A_t - \mathrm{mean}_{t'}(A_{t'})}{\mathrm{stdev}_{t'}(A_{t'}) + \epsilon_{std}}$$

のように計算され、 $\mathcal{L}_{value}$ は

$$V_{clip}(s) = \mathrm{clip}\left(V_\theta(s) - V_{\theta_{old}}(s), -\epsilon_v, \epsilon_v\right) + V_{\theta_{old}}(s)$$

$$\mathcal{L}_{value} = \max\left(\left(V_\theta) - V_{target}\right)^2, \left(V_{clip} - V_{target}\right)^2 \right)$$

と計算される。

本研究では、 V_{target} として $V_{target} = A_t + V_{\theta_{old}}$ を用いた。これは OpenAI Baselines [45] の実装と同じである。

4.3 実験: Mini-RTS における自己対戦の勝率

4.3.1 実験の内容

通常の自己対戦および NFSP を用いて Mini-RTS の均衡解が得られるかを確認するために、これらの手法を用いてプレイヤを学習させ、ルールベース AI に対する勝率を計測する実験を行った。Mini-RTS には AI-Simple と AI-Hit-and-Run の 2 種類のルールベース AI が用意されており、前者は単にユニットを一定数生産して攻撃するだけだが、後者はより複雑な戦略を用いるようになっている。人間のプレイヤにプレイさせた結果、それぞれに対して 90%、50% の勝率であったことが報告されている [31]。

このゲームは二人零和対称ゲームなので、もしプレイヤがナッシュ均衡戦略の一部であるような戦略を取っていれば、そのプレイヤはどんなプレイヤにも搾取されず、必ず 50% 以上の勝率を得る。ゲームが十分小さければ、戦略がどれだけナッシュ均衡戦略に近いかを表す可搾取量を計算することができる [5] が、 Mini-RTS は状態量が多すぎるため、このような評価手法は適用できない。可

搾取量の近似や下界を与えるアルゴリズムは他にも存在するが [46]、上界を計算することはできず、Mini-RTS のような十分大きいゲームで戦略がナッシュ均衡に近づいていることを確実に評価する手法は存在しない。本研究では、単に評価対象となる戦略をルールベース AI と対戦させ、その勝率を評価指標とした。この値は可搾取量の下界の推定量になっている。

4.3.2　実験設定

実験はすべて、以下の設定で行った。

バッチサイズは 128 とし、時間方向のバッチサイズは 50 とした。アルゴリズム 4 の 18 行目の reservoir sampling では、一度に 512 個のデータをサンプルした。フレームスキップは 50 に設定した。すなわち、各プレイヤは 50 フレームごとに行動を選択するようにした。各 Γ_i では 512 ゲームを並列に実行した。ニューラルネットワークのモデルとしては、64 チャンネル、 3×3 カーネルの畳み込み層と $\alpha = 0.1$ の leaky ReLU を組み合わせたレイヤを 4 枚並べ、 2 枚ごとに 2×2 max pooling レイヤを挟むことで body block を構成した。すべてのモデルはこの body block に適切なヘッダ π_{SL} 、 π_{RL} 、または V_{RL} をつけた構造になっており、ヘッダはすべて 1 層の全結合層からなっている。ただし、$\pi_{\cdot}$ には確率分布を表すためにソフトマックス層が付いている。Body block のパラメータは、π_{RL} と V_{RL} を共有させ、 π_{SL} は別のパラメータを用いた。また、プレイヤ間ではすべてのネットワークを共有した。最適化には勾配を 0.5 でクリッピングした確率的勾配降下法を用いた。学習率として強化学習のモデルには 0.01 を、教師あり学習のモデルには 0.001 を用いた。式 (2.1) の計算に用いるハイパーパラメータは、$\alpha = 0.01$、$\beta = 0.5$、$\gamma = 0.99$、$k = 0.95$、$\epsilon_{std} = 10^{-8}$、$\epsilon_v = 0.1$ を用いた。

4.3.3　実験の結果

図 3.1 に実験の結果を示す。AI-Simple に対する勝率で比較すると、対戦相手を AI-Simple に固定して学習させた PPO プレイヤが最も高い勝率を得ているが、このプレイヤは AI-Hit-and-Run に対する勝率が非常に低くなっており、学習時の対戦相手に特化した戦略を学習したことで搾取されやすい戦略になってしまっていることがわかる。AI-Hit-and-Run に対して学習させた PPO プレイヤも同様に、学習時の相手に対しては最も良い勝率を出すことができるが、それ以外の相手には搾取されてしまうことがわかる。

NFSP を用いて学習させたプレイヤは、学習は他の手法と比べて遅いものの、どちらのルールベース AI に対する勝率も学習が進むにつれて上がっていることがわかる。このことから、単純な PPO のような特定の相手に特化した戦略ではなく、より搾取されにくい、ナッシュ均衡に近い戦略が得られていると考えられる。

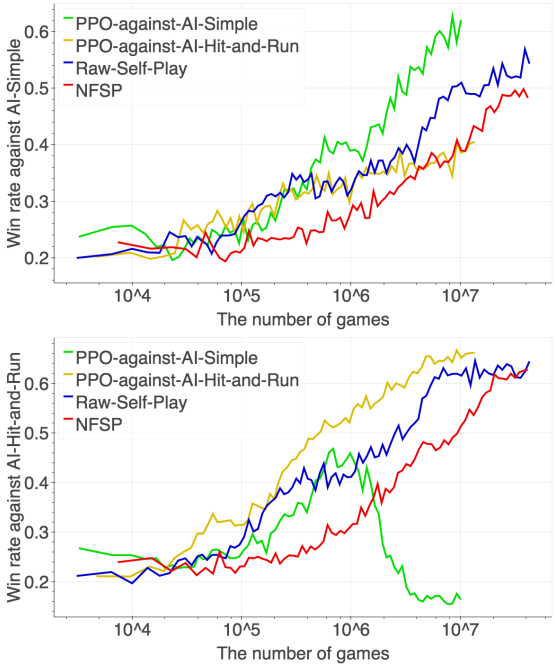

図 3.1: Mini-RTS における各プレイヤの勝率。横軸は学習に用いたゲームの数を対数で示している。上図が AI-Simple との対戦結果を、下図が AI-Hit-and-Run との対戦結果を示している。

また、通常の自己対戦を用いて学習させたプレイヤは、AI-Hit-and-Run に対する勝率で NFSP より早く同じ結果に到達しており、AI-Simple に対する勝率では NFSP よりも高い勝率を得ることに成功している。NFSP は最適応答戦略と平均戦略の両方を同時に学習する必要があるのに対して、通常の自己対戦では最適応答戦略のみを学習すれば良いので、もし通常の自己対戦もナッシュ均衡戦略に収束するのであれば NFSP よりも速く収束することが期待される。通常の自己対戦では戦略が収束する理論的な保証はないが、戦略が収束すればその戦略はナッシュ均衡であるため、実験的には NFSP よりも良くなることがある。実際に自己対戦が振動して収束しない例も報告されている [41] が、今回の実験ではそのような結果は観測されなかった。

自己対戦でない通常の強化学習アルゴリズムを用いて学習させる際に、Tian らは AI-Simple と AI-Hit-and-Run を 50% ずつ混ぜ合わせた AI を対戦相手として学習させることで性能が向上することを示した [31] が、今回はこの実験は行わなかった。本論文の目的は任意の相手に対して搾取されないプレイヤを構成することであり、そのためには少なくとも 1 つ学習時に見ることのできない対戦相手を用意する必要があるからである。

図 3.1 から分かる通り、 PPO を用いて特化した戦略を学習させた場合でも、勝率は 65% 程度に留まっている。これは、 Mini-RTS がランダム要素の強いゲームであることに起因する。Mini-RTS では、ゲーム開始時に、リソースや基地の位置、ユニットなどがお互いのプレイヤにランダムに配ら

表 3.1: 各プレイヤに対して PPO を用いて最適応答戦略を計算させたときの PPO プレイヤの勝率。† が付いている結果を除いて、いずれも 10^7 ゲームの学習を行った。

Agent	Win rate
Random	0.71
AI-Simple	0.62
AI-Hit-and-Run	0.65
PPO against AI-Simple	$> 0.80^{\dagger}$
PPO against AI-Hit-and-Run	0.56
Raw self-play with PPO	**0.45**
NFSP with PPO	**0.44**

れる。このユニットの数などは非対称であり、かつフレームスキップが大きいため、もしゲーム開始時に相手プレイヤが多数のユニットを持っており、自分が一切ユニットを持っていなかった場合、プレイヤは相手プレイヤの攻撃をどうやっても防ぐことができない。実際、純粋なランダムプレイヤに対して PPO を用いて 10^7 ゲーム学習させたプレイヤを用意し、簡易的な実験を行ったところ、同じランダムプレイヤに対する勝率は 71% に留まってしまい、十分学習させたプレイヤであっても 29% のゲームでランダムプレイヤに敗北してしまうという結果が得られた。

4.3.4　可搾取量の近似的計測

実験で得られたプレイヤをより詳細に分析するため、別の PPO プレイヤを用意し、実験で得られたプレイヤに対して 10^7 ゲーム対戦させて学習させ、勝率を計測した。もし PPO が最適な戦略に収束すれば、この勝率は対象となるプレイヤの、不完全記憶かつフレームスキップがある状況下における可搾取量を表すことになる。

結果を表 3.1 に示す。他のプレイヤと比較して、自己対戦及び NFSP を用いて得られたプレイヤが搾取されにくい戦略になっており、 PPO が 50% 未満しか搾取できていないことがわかる。この結果は、これらのアルゴリズムによって得られた戦略が、不完全記憶かつフレームスキップがあるような任意の戦略に対して搾取されないことを示唆している。

4.3.5　得られた戦略の観測

得られた戦略が具体的にどのようなものなのかを確かめるため、 NFSP によって得られたプレイヤとこれに対して学習させた PPO プレイヤの対戦結果のスクリーンショットを図 3.2 に示した。この対戦では、 NFSP プレイヤはまず 2 体の近接戦車を建設し、1 体の遠隔戦車を建設し、その後に一斉攻撃を仕掛けている。この行動は人間にとって合理的な挙動である。なぜなら、ゲームの性質

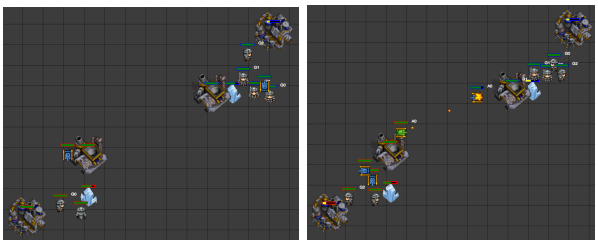

図 3.2: NFSP プレイヤ（左下赤枠）と、それに対して学習させた PPO プレイヤ（右上青枠）の対戦のスクリーンショット。青色の戦車は近接戦車を、緑色の戦車は遠隔戦車を表す。NFSP プレイヤは（画像左）まず近接戦車を建設し、（画像右）次に遠隔戦車を建設した。

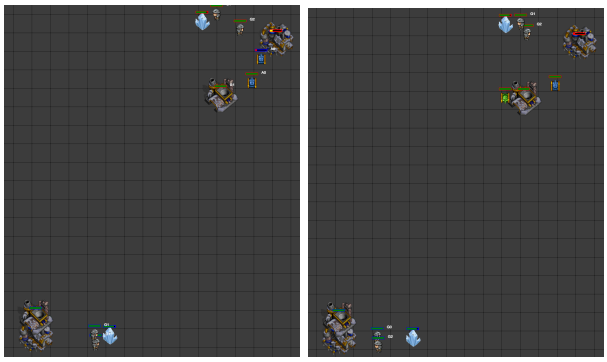

図 3.3: NFSP プレイヤ（右上赤枠）と、それに対して学習させた PPO プレイヤ（左下青枠）の対戦のスクリーンショット。（画像左）PPO プレイヤの戦車によって NFSP プレイヤの基地が攻撃されているが、かろうじて耐えている。（画像右）相手の攻撃を耐えた直後、 NFSP プレイヤは遠隔戦車を建設した。

上、近接・遠隔戦車はそれぞれ防御・攻撃に優れており、 fog-of-war によって相手がいつ攻撃してくるかわからない状況下では、まず防御ユニットを建設してから攻撃ユニットを建設することで相手の攻撃に備えつつ攻撃の準備を行うことができるからである。

図 3.3 に別の例を示す。この例では、 Mini-RTS のランダム性により、 ゲーム開始時点で PPO プレイヤには兵舎がある一方で、 NFSP プレイヤには兵舎がないという大きな差が開いている。 NFSP プレイヤが兵舎を建設している間に、PPO プレイヤは兵舎を使って戦車を建設し、 NFSP プレイヤに攻撃を仕掛けてきている。しかし、 NFSP プレイヤは防御に優れた近接戦車ではなく、攻撃に優れた遠隔戦車を新たに建設し、相手プレイヤにカウンターを仕掛けることでこの試合に勝利している。今回の実験設定ではすべての命令が大域的であるため、相手が攻撃を仕掛けてきた時点で相手は他に防衛のための戦力を残していないことがわかる。NFSP プレイヤはこのことを利用してカウンターを成功させている。

4.4 実験: 自己対戦による NFSP の事前学習

4.4.1 アイディア

前節において、 NFSP は搾取されにくい戦略を得られるものの、他の手法に比べて学習が遅いという問題点が観測された。これに対して、 PPO を用いた通常の自己対戦は、速く収束し特定の相手に依存しない戦略が得られるものの、収束する理論的保証がないという問題点があった。もし自己対戦で学習したプレイヤを用いて NFSP プレイヤの事前学習ができれば、双方の問題点を打ち消すことができると考えられる。

この考え方は、不完全情報ゲームの解を求める手法の一つである CounterFactual Regret Minimization+ (CFR+) [20] にも用いられている。CFR+ は regret matching アルゴリズムの一種であり、 regret を最小化する戦略を求める操作を繰り返すことで、その戦略の平均がナッシュ均衡戦略に収束するというものであるが、 CFR+ ではこの平均操作を遅延させ、途中から平均を取り始めることで性能を向上させている。自己対戦を用いて NFSP を事前学習させることは、この平均戦略の計算を遅延させることと同じであると考えることができる。

4.4.2 強化学習部分の事前学習

まず、NFSP の強化学習のパラメータを通常の自己対戦によって事前学習させ、勝率を計測した。すなわち、PPO による通常の自己対戦を用いてまず学習を行い、その学習で得られたパラメータを用いて NFSP の強化学習部分のモデルを初期化し、そのモデルを用いて NFSP の学習を行った。この際、教師あり学習部分のモデルは通常通り初期化を行った。

結果を図 4.1 に水色の線で示す。学習は事前学習を行わない NFSP と比べて多少速くなっているものの、 NFSP プレイヤの性能を大きく向上させることはなかった。これは、 NFSP が教師あり学習と強化学習の両方を同時に学習する必要があることに起因すると考えられる。NFSP の強化学習部分は自己対戦によって事前学習されているため、学習初期の段階で強化学習はある程度「良い」戦略を持っている。しかし、教師あり学習はランダムに初期化されているため、教師あり学習が強化学習の確率分布を学習するまでは強化学習はランダムプレイヤと対戦することになり、その間に事前学習が忘却されてしまう。これによって結果が向上しなかったのではないかと考えられる。

4.4.3 教師あり学習部分も含めた事前学習

この問題を解決するため、自己対戦で学習したパラメータを、強化学習だけでなく教師あり学習のパラメータにも転用した。教師あり学習の確率分布を出力するモデルと強化学習の確率分布を出力するモデルは同一であるため、この操作は単に π_{RL} のパラメータを π_{SL} にコピーするだけで行うことができる。これは強化学習のアルゴリズムが policy based であるときのみ行える操作である。

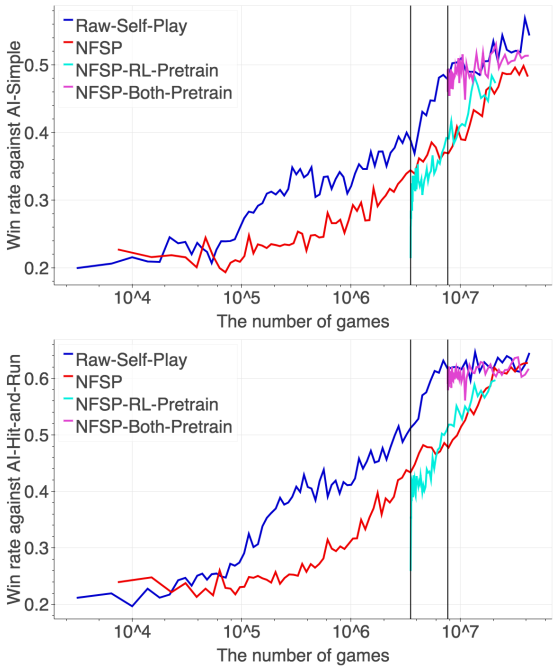

図 4.1: 事前学習を用いた NFSP プレイヤの勝率。横軸は学習に用いたゲームの数を対数で示している。上図が AI-Simple との対戦結果を、下図が AI-Hit-and-Run との対戦結果を示している。黒線は事前学習から NFSP の学習に切り替えたタイミングを表している。

この実験の結果を、図 4.1 に紫色の線で示す。通常の自己対戦と比べると結果は下がっているものの、 NFSP は事前学習した結果を崩壊させることなく維持し、強化学習のみを事前学習させた場合と比べても性能を向上させることに成功している。この結果は、学習は速いが収束の保証はない自己対戦を用いて、学習は遅いが収束を保証する理論的背景がある NFSP を事前学習させられることを示唆している。

第5章　結論

5.1　本研究のまとめ

本研究では、大規模な不完全情報ゲームにおけるナッシュ均衡戦略を求めるために、 ゲーム理論の手法を機械学習の手法で近似することで、事前知識なしにナッシュ均衡戦略を安定して求められることが期待される、 NFSP という手法について研究を行った。

まず、この手法と他の手法を比較し、本研究の目的であるナッシュ均衡戦略を求めるにあたって、NFSP が手法として妥当かを検討した。その結果、ニューラルネットワークを用いた手法である NFSP が FSP と比較して不安定であることから、アルゴリズムに改善の余地があると判断した。

次に、 NFSP の問題点と思われる部分を複数指摘し、その問題点を改善するための手法を提案した。それらの手法について比較実験を行ったところ、性能が大幅に改善するということはなかったものの、元の手法と比較しても遜色のない結果が得られた。

さらに、 NFSP に用いられる強化学習の off-policy 性が活かせないことを指摘し、 NFSP を方策勾配法と組み合わせる手法を提案した。その提案手法を非自明な不完全情報ゲームである Mini-RTS というゲームに適用し、通常の強化学習を用いた自己対戦の手法と定量的な比較を行った。その結果、通常の強化学習より学習は遅いものの、提案手法が実際に搾取されにくい戦略を獲得できることを示した。

最後に、実験の結果得られた考察から、自己対戦を用いて事前学習を行うことで NFSP の学習速度を向上させる手法を提案し、その手法を実際に適用して評価を行った。その結果、学習されたプレイヤ自体の性能向上はほとんど見られなかったものの、提案手法は事前学習の戦略を崩壊させることなく維持しており、提案手法の有効性が示唆された。

5.2　今後の課題

本研究では、 NFSP における強化学習のコントローラーに、 LSTM などの記憶を保持するネットワークを用いなかった。これは、LSTM などを用いると実験にかかる時間が大幅に増加してしまい、試行錯誤ができなくなると考えたためである。しかしながら、第 4.2.2 項でも述べたとおり、本来 NFSP は完全記憶ゲームの上で成り立つ手法である。実際、第 2 章でもゲームが完全記憶ゲームであることを何度も仮定した。このため本研究では、 RTS ゲームに適用して得られた戦略が、どの

程度ナッシュ均衡戦略から離れているかについて深く議論することができなかった。この点をより精査し、学習時間や replay buffer のサイズを損なうことなく完全記憶性を担保する手法について調べたい。

また、本研究を行っている途中に、Deep CFR [47] という手法が提案された。この手法は NFSP と同様にニューラルネットワークを用いて不完全情報ゲームのナッシュ均衡戦略を求める手法であるが、 FP ではなく CFR を元に構成されている。この手法は reservoir sampling を 2 回行っており、実際に大規模な不完全情報ゲームに適用するのは困難であるように思われる。また、サンプリングの手法としてゲームの遷移規則を用いた手法を採っており、筆者はこの部分について任意のサンプリング手法を用いることができると主張しているが、実際にゲームの遷移規則を用いない手法を適用した場合には性能は大きく落ちてしまうと考えられる。しかしながら、 CFR は FP に比べて非常に効率よくナッシュ均衡戦略を求められる手段であり、これにニューラルネットワークを用いて関数近似が行えることは非常に画期的である。また、サンプリングの手法についても Variance Reduction in MCCFR [48] を用いて分散を小さくすることで、性能を落とすことなくゲームの遷移規則を用いない手法にすることができる可能性がある。これらについてより深く検討し、実験等を行いたい。

参考文献

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, pp. 529–533, 2015.

[2] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

[3] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv:1603.01121*, 2016.

[4] H. W. Kuhn. Extensive games and the problem of information. In *Contributions to the Theory of Games*, Vol. 2. Princeton University Press, Princeton, N.J., 1953.

[5] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 258–265. AAAI Press, 2011.

[6] Nick Abou Risk and Duane Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1, pp. 159–166. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[7] 河村圭悟, 水上直紀, 鶴岡慶雅. 未知環境における多人数不完全情報ゲームの戦略計算. ゲームプログラミングワークショップ 2016 論文集, pp. 188–195, 2016.

[8] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, Vol. 13, No. 1, pp. 374–376, 1951.

[9] David S. Leslie and E.J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, Vol. 56, No. 2, pp. 285 – 298, 2006.

[10] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 805–813. JMLR Workshop and Conference Proceedings, 2015.

[11] Amy Greenwald, Jiacui Li, Eric Sodomka, and Michael Littman. Solving for best responses in extensive-form games using reinforcement learning methods. *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 2013.

[12] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, Vol. 11, No. 1, pp. 37–57, March 1985.

[13] J. S. Shamma and G. Arslan. Dynamic fictitious play, dynamic gradient play, and distributed convergence to nash equilibria. *IEEE Transactions on Automatic Control*, Vol. 50, No. 3, pp. 312–327, March 2005.

[14] Peter Jin, Kurt Keutzer, and Sergey Levine. Regret minimization for partially observable deep reinforcement learning. In *Proceedings of the international conference on machine learning*, 2018.

[15] Christopher J.C.H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, Vol. 8, No. 3, pp. 279–292, 1992.

[16] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[17] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[19] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, pp. 1729–1736. Curran Associates, Inc., 2008.

[20] Oskari Tammelin. Solving large imperfect information games using CFR+. *arXiv:1407.5042*, 2014.

[21] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, pp. 837–846. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[22] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, Vol. 347, No. 6218, pp. 145–149, 2015.

[23] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22*, pp. 1078–1086. Curran Associates, Inc., 2009.

[24] Harold W Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, Vol. 1, pp. 97–103, 1950.

[25] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pp. 550–558, Arlington, Virginia, United States, 2005. AUAI Press.

[26] Keigo Kawamura, Naoki Mizukami, and Yoshimasa Tsuruoka. Neural fictitious self-play in imperfect information games with many players. In *Computer Games Workshop at IJCAI-17*, 2017.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[28] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48, pp. 1928–1937, 20–22 Jun 2016.

[29] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. In *Advances in Neural Information Processing Systems 30*, pp. 5392–5402, 2017.

[30] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John

Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv:1708.04782*, 2017.

[31] Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and C. Lawrence Zitnick. ELF: An extensive, lightweight and flexible research platform for real-time strategy games. In *Advances in Neural Information Processing Systems 30*, pp. 2659–2669, 2017.

[32] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv:1712.01815*, 2017.

[33] Santiago Ontanon. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 58–64, 01 2013.

[34] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv:1611.00625*, 2016.

[35] S. Ontan, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 5, No. 4, pp. 293–311, Dec 2013.

[36] Glen Robertson and Ian Watson. A review of real-time strategy game AI. *Ai Magazine*, Vol. 35, pp. 75–104, 12 2014.

[37] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé, III. Opponent modeling in deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, Vol. 48, pp. 1804–1813. JMLR.org, 2016.

[38] Tim Baarslag, Mark J. C. Hendrikx, Koen V. Hindriks, and Catholijn M. Jonker. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, Vol. 30, No. 5, pp. 849–898, Sep 2016.

[39] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, Vol. 12, No. 4, pp. 1–15, 04 2017.

[40] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1021–1026, 2001.

[41] Oleg Klimov and Jhon Schulman. Roboschool. `https://blog.openai.com/roboschool/`, 2017. Last visited 2018-08-29.

[42] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1146–1155, 2017.

[43] Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for AAMAS, 2018.

[44] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs. In *Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[45] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017. Last visited 2018-08-29.

[46] Viliam Lisy and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI) Workshop on Computer Poker and Imperfect Information Games*, 2017.

[47] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. *arXiv:1811.00164*, 2018.

[48] Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. *arXiv:1809.03057*, 09 2018.

発表文献

本研究に関する発表文献は以下の通りである。

- 河村圭悟, 鈴木潤, 鶴岡慶雅. 未知環境における多人数不完全情報ゲームの戦略計算. ゲームプログラミングワークショップ 2017 論文集, pp. 80-87, 2017.
- 河村圭悟, 鈴木潤, 鶴岡慶雅. Neural Fictitious Self-Play における探索由来のデータを含めない教師あり学習による性能改善. 人工知能学会全国大会論文集, 2018.
- 河村圭悟, 鶴岡慶雅. 自己対戦を用いた Mini-RTS の均衡戦略の求解. ゲームプログラミングワークショップ 2018 論文集, pp. 114-119, 2018.
- Keigo Kawamura and Yoshimasa Tsuruoka. Neural Fictitious Self-Play on ELF Mini-RTS. AAAI-19 Workshop on Reinforcement Learning in Games, 2019.

また、その他の発表文献は以下の通りである。

- Keigo Kawamura, Naoki Mizukami, and Yoshimasa Tsuruoka. Neural Fictitious Self-play in Imperfect Information Games with Many Players. In Computer Games Workshop at IJCAI-17, 2017.

謝辞

本研究を行う上で、多くの方にお世話になりました。

指導教員である鶴岡慶雅教授には、研究内容について様々な面でご指導いただき、深い議論をさせていただきました。先生の時間を奪ってしまうのは申し訳ないという思いもありつつ、自分の悩んでいたことが先生に相談してすぐに解決するといったことが何度もあり、また、私の突拍子もない思いつきや疑問に対しても適切な関連論文を提示してくださり、その度に尊敬の念を深めておりました。自分の研究内容に自信が持てなくなったこともありましたが、先生が太鼓判を押してくださったことで研究を続けることができました。本論文を出すことができたのは先生のおかげです。本当にありがとうございます。

また、博士課程の先輩であった亀甲博貴さん、橋本和真さん、江里口瑛子さん、水上直紀さんには、研究の面で多くのアドバイスをいただきました。特に亀甲先輩と水上先輩には、私が学部 4 年の頃からこの分野の基礎知識を身につけるにあたって多くの疑問に答えていただき、かつ分からない点についてホワイトボードの前で何度も議論をさせていただき、大変助かりました。修士論文のテーマの元になった卒業論文の内容は元々水上先輩のアイディアを膨らませたものであり、その点でも水上先輩には大変お世話になりました。修士課程の先輩であった扇本岳大さん、衣川和尭さん、田口直弥さん、徐揚さん、また学部時代の先輩であった村上優樹さんにも、この場を借りて御礼申し上げます。自分が研究を行う上で、先輩方の研究やそのスタイルを大いに参考にさせていただきました。

同期の水谷陽太君には、主にプログラミングの設計パターンの話から細部に至るまで、あるいは論文の内容や研究の方向性に関して、はたまた研究とは全く無関係な雑談について、様々な面で楽しく深い議論ができ、大変お世話になりました。水谷君は私の持っていない知識を数多く持っているので、雑談等を通して私も知見を広げることができました。研究を楽しく続けていけたのは水谷君のおかげです。ありがとうございます。

後輩や研究生の皆様にも、（先輩・同期に比べて人数が多いため、一人ひとり名前を挙げることはしませんが、）大変お世話になりました。研究の熱い議論に参加させてもらったり、少し疲れたときに雑談に付き合ってもらったり、あるいは研究室で一緒に遊んだり、多くの経験がありました。後輩の皆様のこれからの研究成果を楽しみにしています。

また、私が金銭面や生活などを気にせず自分のペースで研究生活を送ることができたのは、学部時代に奨学金を貸与してくださった公益財団法人みずほ育英会様、修士 1 年次に奨学金を賞与してくださった株式会社トヨタ自動車様および株式会社ドワンゴ様、そして何よりこれまで支え育てて

くださった両親と親戚の皆様のおかげです。この場を借りて御礼申し上げます。特に両親には、修士課程に進むことで負担も心配も掛けてしまいましたが、それでも修士課程に進学させてくれたことに大変感謝しています。いつもありがとうございます。

この修士論文を読んでいるあなたがどなたか存じ上げませんが、本論文を読んでいただいてありがとうございます。拙い部分は多々ありますが、少しでも何かのお役に立てていれば幸いです。修士論文を読む人は研究室に入った初学者が多いと思うので、特に第 2 章では前提知識を用いず平易な表現を心がけたつもりです。何かわからない点がありましたら、遠慮なく直接連絡して聞いて下さい。ありがとうございます。