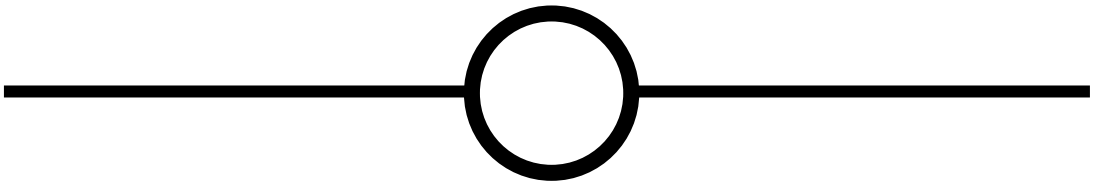
# PokéAI

～人工知能の考えた最強のポケモン対戦戦略～

[#1：初代 1vs1 編]



- ポケモンバトルシミュレータの実装
- 深層強化学習フレームワークとの連携
- 技構成の離散最適化
- 技選択の強化学習

@select766 / ヤマブキ計算所

# まえがき

この薄い本は、テレビゲーム「ポケットモンスター」（以下ポケモンと表記、開発：ゲームフリーク）の対戦ゲームとしての部分に焦点を当て、その戦略を人工知能（AI）に考えさせたらどうなるか、というテーマについて書かれたものです。ポケモンは 1996 年に任天堂の携帯ゲーム機「ゲームボーイ」用ソフトとして初代バージョン（赤・緑）が発売され、その後 20 年以上にわたり続編・派生ゲーム・テレビアニメ等が展開されています。ポケモンの楽しみ方はさまざまですが、そのひとつとしてゲーム中のポケモンバトル（対戦）の戦略を考えるというものがあります。ポケモンバトルは、ポケモン同士が対面し、技を出し合って相手のポケモンを倒したほうが勝ちというゲームです。初代でも、ポケモンの種類は 151、技の種類は 165 種類あり、これらをどう組み合わせて戦うかを考えるのは非常に面白いテーマとなっています。発売翌年の 1997 に任天堂公式大会が開かれ、インターネットが普及した現代ではオンライン対戦が盛んにおこなわれています。

さて、ポケモンを遊んだことがある方なら「どんなポケモンや技が最強なのか」ということが気になるのではないでしょうか。じゃんけんのように相性があるため唯一の最強が決まるわけではありませんが、トップクラスの強さの戦略というのは確実に存在するでしょう。これを考えるのがゲームの醍醐味のひとつといえますが、筆者はこれを人工知能に考えさせたらどんな戦略を導き出すのかということに興味を持ちました。

ゲームにおける人工知能といえば、2015 年に AlphaGo という囲碁 AI がトップ棋士を凌駕したというニュースは記憶に新しいところです。AlphaGo のプログラムには、どんな戦略が強いかという情報は入力されていません。プロ棋士の対戦記録（棋譜）や AlphaGo が自分自身と対局することで得られた情報から、自ら戦略を編み出しました。その結果として、人間が思いつかなかった新たな戦略を披露し、囲碁界に衝撃をもたらしました。ポケモンバトルでも同様の人工知能技術によって新たな戦略が生み出せないでしょうか？人工知能という広い分野の中では、ポケモンバトルと囲碁は比較的近い技術を適用可能と考えられますが、ポケモンバトルのルールは囲碁のルールとはさまざまな点で異なり、同じ技術で戦略を生み出せるわけではありません。本書の研究の目的は、ポケモンバトルの世界で人工知能技術を応用し、新たな戦略を生み出す手法を開発することです。まだ課題に取り掛かったばかりで強力な戦略を生み出すには程遠い状況ではありますが、ここまで

のまとめを本書で提示したいと思います。

## 準備編との違い

本書の関連書として、2017 年末に「準備編」として異なる条件での研究結果を発表しております。準備編ではポケモン 9 種類、技 20 種類のみ存在する環境での実験を行いました。今回の条件ではポケモンが 151 種類に増えるなど、考慮すべきパターンが大幅に増えているため手法を一部変更しております。内容が共通する箇所は文章を流用・修正し、本書だけで完結するよう編集してあります。

## 対象読者

本書は、ポケモンの本編ゲーム [*1]を一度は遊んだことがある方を対象としています。本書では初代バージョン（赤・緑）のルールに準拠して記述しますが、サン・ムーン等最新のバージョンだけを遊んだことがあるという方でもほとんど問題ありません。すべての技の効果を把握している必要はまったくありませんが、ポケモンバトルの流れ、それを有利に進めるためのタイプ相性の概念等を理解していることを前提とします。一方で、人工知能に関する知識は前提としません。数式はほとんど出ませんし、手法部分は読み飛ばして最終的に出てきた戦略だけを眺めていただいてもよいかと思います。

[*1] ポケットモンスター赤・緑（1996）、金・銀（1999）…ウルトラサン・ウルトラムーン（2017）等。ポケモンスナップ等の派生ゲームはルールがまったく異なります。

# 目次

# 第 1 章

# イントロダクション

本書では、ポケモンバトルの戦略を人工知能に考えさせるための技術を開発することを目的とします。

## 1.1 ゲーム AI と人工知能技術

テレビゲームと人工知能は切っても切り離せない関係にあります。インベーダーゲームのような初期のゲームであっても、ただボタンを押したら弾が発射されるというルールが動作するだけでなく、敵がプレイヤーを倒そうと賢くふるまうからこそゲームとしての面白さが生まれるといえます。これは非常に原始的な例ではありますが、ゲームで敵キャラクターのふるまいを決定する機構は人工知能の一例とみなすことができるでしょう。ポケモンのゲーム中においても、野生のポケモンは技をランダムに選択して攻撃してくるのに対し、トレーナーが繰り出すポケモンはこちらにより多くダメージを与える技を選択するようになっている場合があり、一種の人工知能が搭載されることでゲーム性が増しているといえます。

これらのゲームに組み込まれている人工知能は「ルールベース AI」と呼ばれ、ゲーム中のあらゆる場面においてあらかじめ人が設計したとおりに動作するものであり、（電子回路でできていることによる反応速度や正確性はさておき）設計者の腕前を超えるような戦略をとってくるものではありません。本書で考えたいのは、トッププレイヤーに勝利したことで近年話題となった囲碁プログラム AlphaGo [*1]のような、自ら戦略を考えて行動する人工知能です。AlphaGo は、コンピュータが作られるはるか昔から多くの人が戦略を考察してきた囲碁というゲームにおいて、新たな戦略を披露しました。そのほかにも、ブロック崩し等の比較的単純なゲームでは、ゲーム画面と得点を与えるだけで自動的に操

[*1] David Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature 529, 484-489, 2016.

作方法を習得するような技術 [*2]も発表されています。

以後、ゲームに関する人工知能はゲーム AI と表記します。

## 1.2 強化学習の概要

ゲーム AI が新たな戦略を自動的に発見し行動するために活用できる人工知能技術として、強化学習（reinforcement learning）が著名です。強化学習は機械学習（machine learning）の一種ですので、まず機械学習について軽く説明します。

機械学習（machine learning）は、データの性質を計算によって分析・表現するための技術です。幅広い分野ですが、ゲーム AI で使われる状況に限っていえば、ゲームの状態 $s$ （state）を入力として次にとるべき行動 $a$ （action）を出力とするような関数（方策と呼びます） $a = \pi(s)$ を自動的に生成するために利用することができます [*3]。インベーダーゲームの例で考えると、 $s$ は敵、自機、弾の座標などの情報が入り、 $a$ は移動方向、弾の発射等の情報が入ります。これらはゲームの種類により、連続値か離散値か、スカラー（数値 1 つ）かベクトルかの違いがあり、適切な表現方法が異なります。

機械学習を使わずルールベース AI として $\pi$ を設計するとすれば、「敵の弾が正面にあれば右に移動」「敵が正面にいるなら弾を発射」などの条件を 1 つ 1 つプログラミングすることになります。しかしながらルールベース AI の強化は大変で、「敵が自機の左側に 3 体いて、右側には 5 体、敵の弾が 2 マス右にあるとき左に移動」というような詳細な条件をすべて想定してプログラミングをする必要が生じます。これはただ根気が必要というだけでなく、ゲームのうまい人でも自分の思考をすべて明確な条件に書き下すことができず限界がすぐに訪れます[*4]。

このような限界を打開する手段として、機械学習を活用することができます。機械学習を用いると、方策 $\pi$ の中身を自動的に調整することができます。$\pi$ の形態を一次式、二次式等にある程度決めたうえで、その中で自動調整するパラメータを $w$ （ウェイト、重みと呼ばれます）と書き、$\pi(s; w)$ と表現することにします。インベーダーゲームのゲーム AI を作る場合を考えます。 $s = (s_1, s_2, s_3)$ 、 $w = (w_1, w_2, w_3)$ というベクトルとしたときに、「敵が自機の左側に $s_1$ 体いて、右側には $s_2$ 体、敵の弾が $s_3$ マス右にある」ときに、$\pi(s; w) = s_1w_1 + s_2w_2 + s_3w_3$ と定義し、 $\pi(s; w) > 0$ なら右に移動、 $\pi(s; w) < 0$ なら左に移動ということにします。そして、機械学習を用いてこの $w$ を自動的に調整します。機械学習の一分野である教師あり学習（supervised learning）では、うまい人にゲームを

[*2] Volodymyr Mnih et al., Human-level control through deep reinforcement learning. Nature 518, 529-533, 2015.

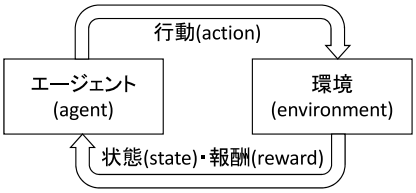
[*3] ここで $\pi$ は円周率とは関係がなく、任意の関数です。強化学習の文脈でよく用いられます。

[*4] 実用分野において、エキスパートシステムの失敗という呼ばれ方をしています。

プレイしてもらいデータを蓄積し、そのプレイを模倣するように $w$ を調整します。例えば $s=(2,5,0)$ のときは右に移動、 $s=(4,3,-2)$ のときは左に移動したという記録があれば、 $\pi((2,5,0),w)>0$ かつ $\pi((4,3,-2),w)<0$ を満たすような $w$ を求めるということになります。ルールベースの場合と違い、人の判断基準を明確に書き下す必要はなく、人の判断結果をもとにパラメータを自動調整する点が大きな違いです。ところで、単純に $s=(2,5,0)$ なら右、 $s=(4,3,-2)$ なら左というデータベースを記憶して一致するものを呼び出す、つまり実質ルールベース AI と同じことをするのではなく、 $w$ というパラメータを学習するという回りくどい操作をしているのには大きな意味があります。データベースだけでは、ゲーム AI がゲームをプレイしている最中に $s=(3,4,-1)$ というような未知の入力が生じたときに、どういう結果を返せばいいか定まりません。機械学習では大量のデータをもとにパラメータ $w$ を学習することにより、未知の入力に対してもある程度正しい結果を返せるようになります。これを汎化能力といい、データの丸暗記とは異なる機械学習の重要な目標です。精度向上のため、 $\pi$ の形態として先述した一次式のほか、 $\pi(s;w)=s_1w_1+s_1^2w_2+s_2w_3+s_2^2w_4+s_3w_5+s_3^2w_6$ のように二次の項を入れるなど、さまざまな関数形（モデル）を試したり、 $w$ の調整手法を変えたりして問題ごとに最適な解き方を考える必要があります。

教師あり学習では、うまい人がいて、そのプレイを模倣するようなパラメータを学習するという考え方を取りました。逆にいえば、ゲームのうまい人がいなければこの手法は成立しません。また、人の模倣なので誰も思い付いたことがない戦略をとるということもできません。このような場合に活用できる手法が強化学習です。強化学習では、コンピュータが方策 $\pi$ に従いゲームをプレイします。そしてゲームの得点をもとに、プレイの方策を調整していきます。極めて単純化すると、ランダムな重み $w^1,w^2$ を用意し、 $\pi(s;w^1)$ と $\pi(s;w^2)$ にしたがってゲームをプレイします。その結果ゲーム終了時点で前者は 100 点、後者は 50 点を獲得したとします。この場合、 $w^1$ のほうが $w^2$ よりもよいパラメータだと推定できます。この考え方を拡張し、何度もゲームをプレイしていく中で高得点を得られるパラメータを自動的に獲得するのが強化学習の考え方です。教師あり学習では人間がお手本となるプレイを見せる必要がありましたが、強化学習ではその必要がなく、コンピュータが自らのプレイを元に学習していくことができるという大きなメリットがあります。一方で技術的には難しく、 $\pi$ から得られる操作 1 つ 1 つが良いか悪いかはわからず、ゲーム全体の得点という大雑把な情報だけから良いパラメータを見つけなければなりません。この難しさを克服するための研究が続けられています。

強化学習はゲーム AI だけでなくロボットの制御などにも用いられるので、より一般的な用語とともに枠組みを図 1.1 に示します。




▲図 1.1　強化学習の枠組み

環境とはゲームのことであり、エージェントとはゲームにおけるプレイヤーのことです。環境からゲームの状態が与えられ、それに基づいてエージェントが行動を選択します。選択した行動に従いゲームが進み、また次の状態がエージェントに送られます。ポケモンバトルでは、行動は技またはポケモンの交換の選択を 1 回行うことに相当し、その結果 1 ターン進んだ後の状態（自分や相手の HP など）が送られてくるということになります。報酬はゲームの得点であり、エージェントの目的は「将来的に」受け取れる報酬の合計が最大となるように行動することが強化学習の目的となります。ポケモンバトル等のゲームは相手に勝利することが目的であり、報酬は勝利・敗北が決まったときにのみ発生し、他のときはどれだけ行動しても 0 しか返ってこないという設定になります。ほとんどの行動に対して報酬が与えられないため正解かどうかわかりづらい一方、それらが良い行動でなければ最終的に報酬が得られないというのが強化学習の難しい点です。強化学習の詳細については、専門書 [*5]をご参照ください。

## 1.3　本書で考える範囲と対象ルール

本書で述べるポケモンバトルとは、初代ポケモン（赤・緑）の対戦部分、とくに通信対戦の環境を指します。また、ゲーム AI は、通信対戦のプレイヤーを人の代わりにコンピュータプログラムで実現することを指します。ポケモンというゲームは RPG でありストーリーがありますが、そこには触れません。なお、ゲーム中に出てくる敵トレーナーの行動を選択する機構も同様にゲーム AI の一種と呼ぶことができます。しかし、これは「相手のポケモンに対して相性が良い技を選ぶ」という程度の単純な思考パターンを組み込んであるルールベース AI にすぎず、設計者が考えた戦略を超えた行動をとることはできません。本書では、フーディンが強いとか、飛行タイプの相手にはふぶきが有効であるといった戦略に関する情報を人がプログラムに組み込むことなく、これらの知識を自動的に獲得し、戦略を構築できるような上位レベルのプログラムを開発します。戦略をプログラミングするのではなく、戦略の見つけ方をプログラミングするということです。

[*5] たとえば、浅田稔ほか「これからの強化学習」森北出版

ポケモンバトルのルールは世代を追うごとに複雑化していますが、今回はもっとも単純なルールで行います。ルールは初代（赤・緑）に準拠し、さらに簡単にするためポケモン1匹だけを使ったバトルとします。技術開発が進むにつれ、ポケモン交換のある3対3、6匹見せ合いの3匹選出という拡張を進めていきたいと考えています。将来バージョンで増える面白い要素として、金・銀バージョンでの持ち物・場の効果（天候）があります。初代ではあまり複数のポケモンの連携ということを考える余地がないため、複数ポケモンによるコンボを人工知能が見出してくれるのかというのが非常に面白いところです。さらに先に進めばルビー・サファイアバージョンでの特性・ダブルバトルがあります。ダブルバトルは場に同時に2匹のポケモンを繰り出すルールであり、2匹の行動を同時に選択する必要があるため選択肢が大きく広がり、そこにどう切り込んでいくかがポイントです。

ポケモンバトルで行うべきことは、パーティの構築とバトル中の行動選択です。パーティの構築は、どんなポケモンを採用し、どんな技を覚えさせるかという作業です。ポケモン151種類、技165種類の中から適切な組み合わせを選ぶことになります。これはバトルの開始前、対戦相手を見る前に行います。バトル中の行動選択は、自分と対戦相手の状況を見ながら、技を繰り出す場合の最大4種類と控えのポケモンに交換する場合の最大5種類、合計9つの選択肢から1つを選ぶ作業です。ただし今回はポケモン1匹のみの利用のため、交換はありません。1ターンごとに1つの行動を選び、ゲーム終了まで何度も行動選択をしていく必要があります。

## 1.4 ポケモンのゲームシステム

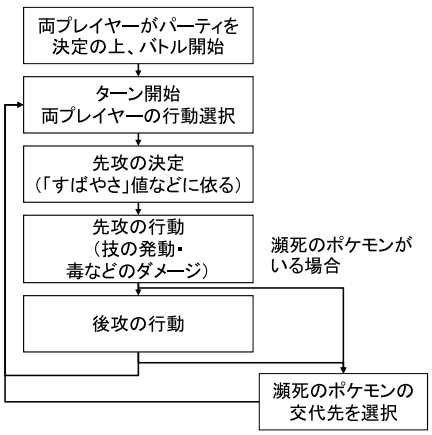
ポケモンバトルの基本的なルールについておさらいしておきましょう。ゲームのバージョンにより少し異なりますが、初代バージョンに準拠します。また、ここで説明しきれないさまざまな例外的状況がありますが、省略しています。

ポケモンバトルは、2人のプレイヤーがそれぞれポケモンを場に出し、技を使って相手プレイヤーのポケモンの体力（HP）を減らし（＝ダメージを与える）、先にすべてのポケモンのHPが0（瀕死状態）になったプレイヤーが負けというルールです。

1人のプレイヤーは1〜6匹のポケモンを持ちます。このポケモンの組をパーティと呼びます。ポケモンは1匹につき最大4種類の技を覚えることができます。ポケモンに覚えさせられる技の候補は4種類より多く、どの技を覚えさせておくかは戦略のある要素です。また、ポケモンの種類により覚えられる技は異なります。たとえば、氷タイプのポケモンであるフリーザーは「ふぶき」を覚えられますが、「かえんほうしゃ」は覚えられません。技を覚えさせたポケモンを組にしてパーティを構築するところまでは、バトルの前に行う作業です。原作ゲームではこの部分でポケモンの捕獲・育成という過程が入りますが、本書ではゲームシステム上実現しうるすべてのパーティを使用できるものとします。

バトルが開始すると、両プレイヤーのパーティの先頭のポケモンが場に出ます。以後、1 プレイヤーにつき 1 匹だけ同時に場にポケモンが出ていて、技を出したり相手の技を受けたりします。

バトルはターンという単位で進行していきます。模式図を図 1.2 に示します。ターンの開始時に、両プレイヤーがそれぞれ行動を選択します。行動は、場に出ているポケモンに技を使わせるか、場に出ているポケモンを控えのポケモンと交換するかの 2 種類があります。技を使う場合、各ターンではポケモンが覚えている技の中から任意の技 1 つを選択して使うことができます（状況によっては選択肢が制限されます)。控えのポケモンとの交換は、瀕死になっていない任意のポケモンを選択して交換できます。両プレイヤーが行動を選択したら、ポケモンの「すばやさ」のパラメータ等に基づき先攻・後攻が判定され、まず先攻の行動が実行され、次に後攻の行動が実行されます。これで 1 回のターンは終了です。ターン途中にポケモンが瀕死になった場合、該当プレイヤーは控えのポケモンを選択して場に出します。あるプレイヤーのすべてのポケモンが瀕死になった時点でバトルが終了します。




▲図 1.2　ポケモンバトルの進行

ポケモンおよび技にはそれぞれタイプという属性が与えられており、たとえば「水」タイプの技が「炎」タイプのポケモンに命中すると、通常よりダメージが 2 倍になります。また、「水」タイプのポケモンが「水」タイプの技を使うと、通常よりダメージが 1.5 倍になります。さまざまな相手の弱点を突けるよう、適切なタイプの技を適切なタイプのポケモンに覚えさせておくことが戦略上非常に重要となります。また、技には相手に直接ダ

メージを与える「攻撃技」だけでなく、ポケモンの状態を変化させる「補助技」（変化技ともいう）があります。補助技の 1 つである「さいみんじゅつ」は、相手のポケモンを眠り状態にします。眠り状態のポケモンは、一定ターン経過して目覚めるまで技を使えなくなります。補助技は多種多様な効果のものが存在しますが、直ちに相手にダメージを与えるのではなく将来的に与えるダメージを大きくする、または自分が受けるダメージを減少させるために用います。攻撃技と補助技の連携もまた戦略上の重要な要素となります。

一般的なルールに基づき説明しましたが、先述のように今回はパーティのポケモンを 1 匹のみとするため、交換の操作はありません。

本書では、ポケモンバトルのルールを独自に実装したシミュレータを用いて戦略の学習・検証を行います。実装の詳細は次の章で解説します。

## 1.5 ゲーム理論上の分類

ゲーム理論におけるポケモンは、「二人零和有限不確定不完全情報ゲーム」に属します。ポケモンのルールに即して説明すると、二人とはプレイヤー二人が参加するということ、零和とは一方が勝ち、他方が負けるということ、有限とはゲーム上で取りうる行動の組み合わせ数が有限であるということ、不確定とは技が当たるか外れるかが乱数で判定されるということを指します。不完全情報とは相手の手持ちポケモンの情報が不明であるということに加え、自分と相手が同時に行動を選択するゲームであるということも含んでいます。類似する有名なゲームとしては、トランプゲームのポーカーも同じ区分となります。最近ポーカーの一部ルールにおいて、人間に勝つゲーム AI が開発されたようです [*6]。ただしポケモンとポーカーはゲーム理論上の区分は同じであるものの、実際上は大きな違いがあり同じ技術が使えるとは限りません。一方で、囲碁は「二人零和有限確定完全情報ゲーム」に属します。ランダムに決まったり、相手だけが知っている情報が存在しないということです。これらの違いが、ゲーム AI を開発するにあたりさまざまな違いを生んできます。

## 1.6 必要な技術

ポケモンのゲーム AI を開発するにあたり、他のゲームの AI で使われているどの技術を利用することができ、または新たに開発する必要があるのでしょうか。

まず、教師あり学習により直接的に各場面での適切な行動を学習することは難しいです。場面ごとの正しい行動を表したデータが入手できないためです。最近のオンライン対

[*6] Matej Moravcik et al., DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. arXiv:1701.01724, 2017.

戦可能なバージョンでは、サービス提供元に対戦データが蓄積されているものと思われますが、筆者にはこれにアクセスする手段がありません。

次に、強化学習による行動の選択はポケモンで利用することができます。ポケモンは、自分が行動を選ぶことで環境が変化していき、最終的に勝敗という報酬が得られるという形式を満たしています。ただし、パーティの構築とバトル中の行動選択という 2 つの段階が存在することがユニークです。確かにパーティの構築も「ポケモンや覚えさせる技を 1 つずつ選ぶ」という行動として定式化が不可能ではないものの、バトル中の行動（覚えている技の中から発動するものを選ぶ）とは明らかに性質が異なるものであり同列に扱うことが難しいといえます。そのため、2 つの段階それぞれに適した手法を用意し、適切に組み合わせるという手順が必要です。

一方で、囲碁等で用いられるゲーム木を用いて将来の局面を予測するという技術はポケモンへの適用が難しいと考えられます。ゲーム木は、自分や相手の取りうる行動を列挙し、それぞれの行動をとったときに将来の局面がどうなるかを表現したものです。ポケモンでは、相手のポケモンが覚えている技などの情報が隠されているため、相手の行動を仮定することが困難です。ポーカー等でも相手の手札が隠されているという点は近いですが、あくまで最初の手札はランダムです。ポケモンの場合はバトルの前のパーティ構築から戦略の自由度があり、取りうる状態がはるかに多いためポーカーでの技術をそのまま利用できるわけではないと考えられます。人間の上位プレイヤー同士だと、相手ポケモンの技構成や次の行動などを仮定して将来の状況を予測する「読みあい」が行われます。この領域に達するにはゲーム木のようなアイデアを取り入れた手法の開発が必要でしょう。

強化学習で得られる行動選択の基準は、原則として「この技を選ぶと 80% 勝利しそう」というようなあいまいなものとなります。学習の過程で微分演算が用いられるため、わずかな入力の差に対して出力が大きく異なるような関数は取り扱いが困難であることが一因です。一方でポケモンにはわずかな差で大きく状況が異なるという場面が多数存在します。一例として、技「みがわり」を用いる場合、自分の HP が最大 HP（バトル開始時の HP）の 1/4（端数切捨て）だけ消費されるため、最大 HP が 4 の倍数だと 3 回しか使えない一方、それ以外なら 4 回使えます。また、素早さというパラメータは相手より 1 でも大きければ必ず先制攻撃できるため、仮想敵となるポケモンより 1 だけ大きな値になるように調整することで大幅に有利になります。強力な戦略を考える場合にはこのような厳密な部分を考慮する場合があるため、一定以上の強さを実現するためには整数値を厳密に処理することができる機構が必要となるのは間違いありません。

## 1.7 研究の意義

本書ではごく一部しかアプローチできていませんが、ポケモンのゲーム AI を強化するための研究を通じて、ポケモン以外にも有用な次のような技術が得られるものと期待できます。

- 多段階のゲームにおけるエージェントの最適化
  - パーティ構築とバトル中の行動選択という 2 つの大きく異なる要素の両方を最適化する技術が必要です。
- 整数問題を扱えるエージェントの構造
  - HP が 4 の倍数かどうか、というような整数値としての処理が必要な場面に対応できる技術が必要です。

また、ポケモンファンに対しては次のような貢献ができると考えられます。

- 人が思いつかなかった戦略の提案
- 特別なルールを仮定した場合の戦略の提案

特別なルールは、たとえば「ニンテンドウカップ'99」という大会で用いられたルールが有名です。この大会では、過去の大会における上位チームが使用していたポケモンを使用禁止にするというルールが設けられました。そのため、それまで注目されていなかったペルシアンなどのポケモンが活用されました。特別なルールを仮定した際にどのような戦略が最適となるのか、プログラムを少し書き換えるだけで検証してみることができ、新たな楽しみ方ができると期待できます。

一方で、ポケモンを大学等（税金で活動している組織）における研究テーマとするには難点もあります。

- ゲームのルールが広く知られておらず、一般的な人工知能関連の学会では理解されづらい
  -「ピカチュウ」のようなキャラクターは世界中で何億人もが知っていそうなものの、ポケモンの対戦の基本概念を理解している人は（特に海外で）非常に少ないことが予想されます。論文の査読者に課題を正しく理解してもらうことが難しいでしょう。
- 必要な技術に対して、ルールが過度に複雑
  - 他分野に応用できるような汎用的な技術開発を目的とするには、ゲームのルールが非常に複雑です。何百種類ものポケモンや技が存在しており、実験に必要な実装や計算負荷が高くなります。

- 一企業のクローズドソースな実装がルールになっているという点でも、学術論文としては扱いづらい側面です。挙動を分析してルールブックを作らなければ、再現性のある研究とは言いづらくなってしまいます。

よく知られたポーカー等のルールを少し変更することでポケモンとほぼ同等の技術課題を作り、大学等での研究テーマとすることは不可能ではないと思います。しかし筆者はポケモンというゲームだからこそモチベーションを感じるため、趣味で細々と研究する道を選んでいます。

# 第 2 章

# システムの構成

本書では、ポケモンのゲーム AI を次のモジュールを開発して実現しました。このシステム全体を PokéAI （ポケエーアイ）と名付けています。

大きく分けて 3 つのモジュールを開発しました。

- シミュレータ
  - ゲーム AI を検討する大前提として、ポケモンバトルのルールを再現するシミュレータを実装します。このシミュレータ上でポケモンバトルを行い、手法の開発や評価を行うこととなります。
- パーティの構築
  - バトルに出すポケモン・技の組み合わせを最適化します。
  - 離散最適化ベースの手法、特に山登り法による手法を提案します。
- 行動の選択
  - バトル中の行動の選択を最適化し、有利に立ち回ります。
  - 強化学習ベースの手法、特に深層強化学習による手法を提案します。
  - シミュレータとの接続においては、OpenAI gym という著名な強化学習の実験環境に従ったインターフェースを実装しました。

プログラミング言語は Python 3.6 を用いました。人工知能研究で広く用いられており、有用なライブラリが多数存在するためです。

ソースコードは github にて公開しています。`https://github.com/select766/pokeai`

## 2.1 シミュレータの開発

ポケモンバトルのルールを再現し、任意のパーティを与えてバトルを開始、ターンごとに技を選んで状況が進展するようなシミュレータを実装しました。ゲームボーイのエミュレータではなく、自前で実装しています。これは人工知能とはあまり関係がなく、地道にダメージの計算式や技の効果を調べて再現する作業となります。仕様の再現に当たっては、ポケモン Wiki https://wiki.ポケモン.com/ 等を参考にしました。

▼リスト 2.1　シミュレータの使用法

```
from pokeai.sim import Field
from pokeai.sim.dexno import Dexno
from pokeai.sim.move import Move
from pokeai.sim.party import Party

# ポケモン 1 匹の生成
poke_1 = PokeStatic.create(Dexno.BULBASAUR, \
    [Move.TACKLE, Move.VINEWHIP, Move.SWORDSDANCE, Move.AGILITY])
poke_2 = PokeStatic.create(Dexno.CHARMANDER, \
    [Move.TACKLE, Move.WATERGUN, Move.AMNESIA, Move.DEFENSECURL])
# パーティの生成
party_1 = Party([poke_1])
party_2 = Party([poke_2])
# シミュレータの生成
field = Field([party_1, party_2])

# 次のターンでの行動を設定
field.actions_begin = [FieldAction(FieldActionType.MOVE, move_idx=0),
                       FieldAction(FieldActionType.MOVE, move_idx=2)]
# 1 ターン進める
field.step()

# 現在の HP を表示
print(field.parties[0].get().hp)
```

作成したシミュレータの使用法をリスト 2.1 に示します。PokeStaticクラスにポケモンの種類・技を指定してポケモンを生成、Partyでパーティにまとめ、2 人プレイヤーのパーティを与えてシミュレータ（Fieldクラス）を生成します。ターンごとの行動をそれぞれ与えて stepメソッドを呼ぶことによりバトルが進みます。ここでは具体的なポケモンや行動をプログラムに書き込んでいますが、これを自動的に決めることが次の節以降の課題となります。

▼リスト 2.2　シミュレータのテストの例

```
# assertEqual により、シミュレータの値と期待される値が一致するかどうかをテストする
self.assertEqual(field.parties[0].get().hp, 152)
self.assertEqual(field.parties[1].get().hp, 146)
field.actions_begin = [FieldAction(FieldActionType.MOVE, move_idx=0),
                       FieldAction(FieldActionType.MOVE, move_idx=0)]

# 1 ターン進める
field.step()

# ダメージ 17 のはずなので、HP が 17 減少しているか確認
self.assertEqual(field.parties[0].get().hp, 152 - 17)
# ダメージ 18 のはずなので、HP が 18 減少しているか確認
self.assertEqual(field.parties[1].get().hp, 146 - 18)
```

タイプ相性、どくどく状態のダメージ、はかいこうせんの反動等こみいった状況が色々あるため、テストを実装しました。リスト 2.2 で示すように、ダメージをあらかじめ計算し、シミュレータの動作がそのとおりの結果になるかということをテストします。ここで問題になるのは、ゲームに乱数が絡むことです。通常はそれがゲームを面白くするのですが、テストの際はランダムだと困ります。そのため、テスト用に乱数を固定する機能を実装しました。乱数にもさまざまな用途があり、ダメージのランダム化、追加効果の発生等いろいろあります。そのためリスト 2.3 のように、乱数生成の際にその理由も与えるようにし、特定の理由の際に特定の値を返すような設定を可能としました。

▼リスト 2.3　乱数の固定

```
# 固定された値が出力される乱数生成器
rng = GameRNGFixed()
field.rng = rng
field.rng.set_field(field)

# 次の急所判定では急所に当たる値を出力
rng.enqueue_const(1, GameRNGReason.CRITICAL, 0)

# シミュレータ内の急所判定
# 上記の設定により、必ず 0 が返るようになる
rng.gen(context.attack_player, GameRNGReason.CRITICAL, 255) + 1 < critical_ratio
```

ちなみに、初代ポケモンはゲームボーイの極めて貧弱な CPU で動作するよう作られたためか、値のオーバーフローなど直観的でない挙動が存在します。しかしながら今回は厳密な再現が目的ではないため、正確に再現していない部分があります。また、「へんしん」「みがわり」等特殊な挙動をする技は実装しておらず、147 種類実装しています[*1]。実装が

[*1] 未実装技一覧：いかり・オウムがえし・カウンター・かなしばり・がまん・からではさむ・きあいだめ・くろいきり・しめつける・しろいきり・テクスチャー・へんしん・ほのおのうず・まきつく・みがわり・ものまね・ゆびをふる・わるあがき

面倒である一方、現状の技術ではその活用法を自動的に学習することが困難と思われるためです。

細かいながら結果に影響しそうな事項を記載しておきます。

- 「ふぶき」の命中率は 90%、追加効果で凍る確率は 30% です。氷タイプ以外の相手を 27% の確率で実質即死にできるということを意味します。初代最強技が容赦なく炸裂します。
- PP（技の使用回数上限）は未実装で、どの技も何度でも使用できます。
- 最後の 1 匹同士での「じばく」「だいばくはつ」は使った側が負けとなります。控えのいない 1 匹だけのバトルなので、いかなる場合でも使うと負けという設定になっています。

## 2.2 パーティの構築

ポケモンバトルは囲碁や麻雀等と違い、対戦相手と出会う前から戦いが始まっています。どんなポケモンを手持ちに加え、どんな技を覚えさせておくのかという「パーティ構築」の部分がきわめて重要となります。

### 2.2.1 ゲームシステムに従ったパーティ候補の生成

戦略的なところ以前に、ポケモンのゲームシステム上どんなポケモンでも好きな技を覚えさせられるわけではありません。ポケモンがレベルアップで覚える技、また技マシンで覚える技のデータベースを作成し、それを用いて特定のポケモンが特定のレベルまでに覚える技を列挙する機構を実装しました。

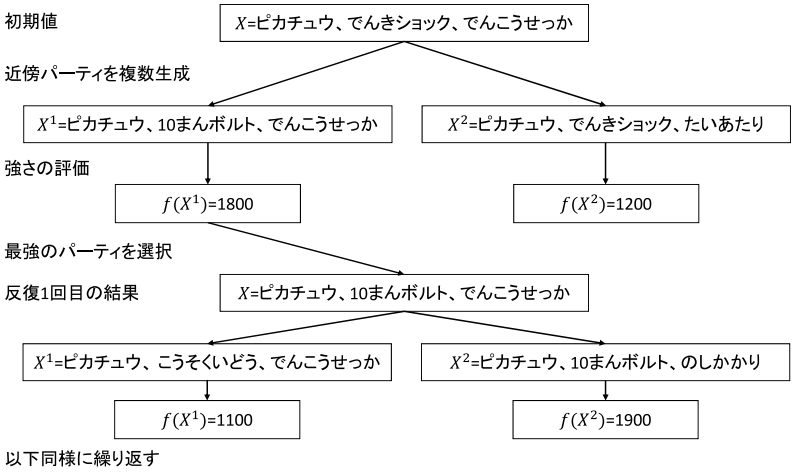
今回のルールでは、レベル 55 のポケモン 1 匹だけを含んだパーティを生成することとしました。レベル 50 でもよかったのですが、カイリューの参加の余地があるようにレベル 55 としました。実験結果からいえば、ふぶき無双のこのルールではまったく採用されなかったというオチが付きます。また、慣例に従いミュウ・ミュウツーは使用不可としました。

### 2.2.2 山登り法によるパーティの構築

パーティの構築は、ゲームシステム上存在しうるポケモン・技の組み合わせのうちもっとも「強い」ものを見つけだす作業だといえます。数学的にいえば、パーティ $X$ に対して関数 $f(X)$ がその強さを表すときに $f(X)$ が最大となる $X$ を見つければよいわけです。このようにパーティの強さを測定する関数を評価関数と呼ぶことにします。強さとは

なんぞやという疑問はさておき、都合のいい評価関数 $f$ が存在したとして $f(X)$ が最大となる $X$ はどうやって見つければいいのでしょうか？ もし $f$ が連続値を入力とし微分可能であれば比較的簡単で、「勾配法」という手法で $X$ を少しずつ良い方向に変化させることができます。しかし今回はそうではありません。 $X$ はパーティ構成なので、連続値ではありません。ポケモンを図鑑番号であらわすとして、25 番のピカチュウに対し 25.01 番のポケモンというのは存在しませんし、24 番のアーボックと近いというわけでもありません。また、前章で紹介した強化学習は「行動ごとに環境が変化していく」という状況で用いるものであり、パーティに取り入れるポケモンや技を 1 つ選ぶことを行動と定義しても、環境からフィードバックが得られるような性質の問題ではないため適用しづらいと考えられます。本書では、離散最適化という枠組みで最強のパーティ $X$ を見つけることを提案します。パーティ構成という変数を考えてみると、ポケモンの変更は一切の連続性がありませんが、技は 4 つ覚えられるのでそのうちの 1 つだけを変えたものは元のものとある程度近く、そして近いパーティは類似した強さをもつという特徴を見出せます。このような状況で $f(X)$ を大きくする $X$ を発見する比較的単純な方法として山登り法が採用できます。図 2.1 に山登り法の概要を示します。最初のパーティ候補「 $X$ =ピカチュウ、でんきショック、でんこうせっか」に対して $f(X)$ を計算しておきます。次に $X$ の近傍として技を 1 つ変更した「 $X^1$ =ピカチュウ、10 まんボルト、でんこうせっか」、「 $X^2$ = ピカチュウ、でんきショック、たいあたり」を考え、 $f(X^1)$ 、 $f(X^2)$ を計算してみます。適切な $f$ があれば、直観的に考えて、 $X^1$ は $X$ や $X^2$ より強いと判断できるでしょう。なので、 $X$ の代わりに $X^1$ を新たなパーティ候補として採用し、同様に近傍を検討、もっとも強いものを採用という操作を反復します。

今回は、山登り法におけるパーティの近傍として技を 1 つ入れ替えたパーティを生成することとしました。技の変更に際して、一定確率で技を 1 個追加する、技を 1 個削除するという操作も含めています。通常の戦略では技の枠 4 つを使い切らないというのはまずありえない話ですが、今回はバトル中の技の選択をランダムに行うという戦略も用います。ランダムに選択する場合、効果の少ない技を持っているよりも強力な「ふぶき」単独のほうが強いという事態がしばしば生じます。ポケモンについては変更できないことにしました。ポケモンによって覚えられる技が異なるため、技を維持したままポケモンを変更できず、ポケモンを変更しようと思うとパーティをまったく別のものに置き換えてしまうことになるためです。




▲図 2.1　山登り法の概要

注意すべきこととして、山登り法は必ずしも $f(X)$ を最大にしてくれる保証はありません。 $X$ の初期値に依存します。特に今回は $X$ の近傍として、技の変更は可能なもののポケモンを変更できないため顕著になります。初期値としてトランセルを選んでしまったらどうにもならないということです。

ポケモンには個体ごとに「個体値」「努力値」という隠しパラメータがあり、同じ種類でも強さのパラメータが異なります。実機ではポケモンをたくさん捕まえたうえで良いパラメータの個体を選ぶ「厳選」が行われますが、本研究ではすべての値を最大に固定しています。実機ではそのような個体を得ることが確率的に困難[*2]であることは注意が必要です[*3]。必ずしも最大値を取ることが最適ではなく、たとえば技「みがわり」を用いる場合は HP が 4 の倍数でないほうが有利という場合があるため、将来的には変更可能なパラメータとなることが望ましいです。

### 2.2.3　強さの測定方法

ここまでパーティ構成 $X$ について議論しましたが、強さを測定する評価関数 $f$ のほうはどうでしょうか。一人プレイのゲームであれば得点が絶対的な基準として使えますが、対戦ゲームでは相手の強さや相性により強さの基準が変わってきます。

相手が単にランダムな相手なら、ふぶきを連打するだけで 90% 勝てるという状況にな

[*2] 16 段階のランダムパラメータが 4 つあり、理想個体が得られる確率は 1/65536

[*3] ルビー・サファイア以降はゲームシステムが変更され、全パラメータを最大にすることはできなくなりました。

ります。しかしそれでは残り 10% に対してどうふるまうのかという点が統計上の誤差（ふぶきが命中するか否かの乱数など）に埋もれてしまいます。相手として極端に弱いトランセルも、ふぶきがほとんど効かないラプラスも等確率に出てきて、どちらに勝っても同じ得点になるためです。これに対処するため、単純な勝率ではなく、イロレーティングを用いることとしました。イロレーティングは対戦ゲームのプレイヤーの強さを数値（レートと呼ぶ）で表す手段の 1 つです。平均的なプレイヤーのレートを 1500 とし、レートの差が 200 あるプレイヤー同士では上位のプレイヤーが 76% の確率で勝利することが期待されます。レートの差が 400 であれば 91% です。このような条件を満たすようにレートを計算することにより、ランダムな相手と戦えば 98% 勝つプレイヤーと 99% 勝つプレイヤーの差をより正確に表すことができると考えられます。また、このレートが近いプレイヤー同士を対戦させることにより、勝敗が明らかな組み合わせを戦わせることによる時間のロスも防げます。

通常のイロレーティングでは、参加する全パーティのレートを 1500 に初期化したのち、レートが近いもの同士を対戦させて勝者のレートを増加、敗者のレートを減少させることを繰り返して収束させます。山登り法で用いる際には、測定対象とは別に、測定基準としてパーティのセットを用意し、セット内での各パーティのレートをあらかじめ計算しておきます。そのうえで測定対象となるパーティがどの程度のレートに位置するのかを計算する手順としています。これは測定基準がランダムにずれないようにすることと、計算時間の短縮のためです。

評価関数としてレートを用いることにしましたが、対戦相手の設定方法としてランダムな相手を与える場合とある程度強い相手を与える場合でも違ってきます。ランダムな相手だとふぶきの連打に対抗できる相手が存在しない可能性が高く、レートが飽和してしまいます。ふぶきを連打してくる相手も含まれる環境でレートを計算することで、強さをより明確に測定可能になると期待できます。

## 2.3 行動選択

パーティを構築してバトルに臨んだ際、バトルの最中行うのは行動の選択です。先述のように、1 ターンごとにポケモンの覚えている技（4 種類）のいずれかを選択することとなります。この部分には、深層強化学習を用いました。深層強化学習は、状態を与えると適切な行動を返すような方策（エージェント）を深層ニューラルネットワーク（deep neural network; 以下 DNN）により表現し、このパラメータを深層学習（deep learning）により最適化することで強化学習を行う手法です。深層強化学習の具体的なアルゴリズムはさまざまなものが提案されていますが、選択できる行動が離散的な（整数で表せる）場

合によく用いられる DQN（Deep Q-Network） [*4] [*5]という手法を用いました。

今回採用する DNN は単純な全結合ネットワークです。次のような式で表現できます。

$$\pi(\boldsymbol{s}; w) = W_3\sigma(W_2\sigma(W_1\boldsymbol{s}))$$

ここで、 $W_1, W_2, W_3$ は実数値の行列で、全部合わせて学習可能なパラメータ $w$ を構成します。$\sigma(x)$ は一種の非線形関数で、活性化関数と呼ばれます。活性化関数として使われる関数は何種類かありますが、ここではもっとも一般的な ReLU $\sigma(x) = max(x, 0)$ を用います。 $x$ がベクトルの場合、要素ごとに独立に計算します。今回、4 種類の技の中から 1 つを選択することになっていますので、 $\pi$ の出力が 4 次元のベクトルになるようにし、値が最大値を示す次元の技を選んでいるとみなします。DNN では、入力に行列を掛けて非線形関数を通すという操作を多段に重ねることにより複雑な関数を表現します。この関数によって、ポケモンバトルの状態を入力として適切な技を選ぶような方策を表現することになります。行列のサイズを大きくするほど、操作の段数が多いほど表現力が高くなりますが、適切なパラメータを学習するためのデータがより多くなります。

DQN アルゴリズムの実装は PFN 社の ChainerRL [*6]というライブラリを用いました。ChainerRL は、深層学習ライブラリ Chainer [*7]を活用し、深層強化学習に必要な機能を提供するものです。構築したパーティを味方側として、ランダムな行動を行うパーティとの対戦を多数行い、その報酬（≒勝率）を最大化するような行動を学習する機構を実装しました。

1 回の学習では、味方側のパーティは 1 つに固定しました。つまり、パーティ構成 1 つに対して 1 つの DNN パラメータ $w$ が学習されます。入力 $s$ に自分のパーティの構成も含めることにすれば、1 つのパラメータであらゆるパーティの行動選択をできて汎用的ではありますが、その分必要なパラメータ数が増え、最適化が困難となります。

### 2.3.1 シミュレータとの接続

シミュレータと強化学習のフレームワーク（ChainerRL）との接続のため、OpenAI gym [*8]という著名な強化学習の実験環境で用いられているインターフェースを実装しました。シミュレータそのものはあくまでゲームのルールを再現するため 2 人のプレイヤーの行動を受け取り 1 ターン進める処理が実装されています。一方で強化学習のためには、

[*4] Volodymyr Mnih et al., Human-level control through deep reinforcement learning. Nature 518, 529-533, 2015.

[*5] 日本語の口語ではよく「ドキュン」と発音される

[*6] `https://github.com/chainer/chainerrl`

[*7] `https://chainer.org/`

[*8] `https://github.com/openai/gym`

1 人のプレイヤーの視点における環境情報の取得、行動の入力、勝敗に基づく報酬の取得が必要となります。

OpenAI gym における環境（シミュレータ）を表すオブジェクトは、 `reset()`と `step()`という 2 つのメソッドを持ちます。

`reset()`は環境を初期状態にするものです。強化学習したいエージェントのプレイヤーのパーティ構成はあらかじめ設定したもので固定し、敵プレイヤー側のパーティは別途設定したパーティのリストから `reset`のたびに新しい 1 つが選ばれます。

`step(action) -> (state, reward, done, info)`は、エージェントの行動を受け取り、技の発動処理を行いゲームを 1 ターン進めるものです。`action`（行動）には、4 種類の技を表す 0~3 の数値を与えます。ポケモンの交換はないので、各数値がどの技を指すのかはパーティが固定されていれば一意に決まります。「そらをとぶ」等複数ターン連続して発動する技では、2 ターン目以降は行動選択を無視します。敵プレイヤーの行動はランダムに選ばれます。将来的には、敵も強化学習結果に基づき行動したほうがよいでしょう。

戻り値の `done`はゲームが終了した場合に `True`となるもので、`info`は未使用です。`reward`（報酬）は、ゲームが終了（どちらかのプレイヤーのポケモンが瀕死）しエージェントが勝利した場合は +1、敗北した場合は-1 となります。ゲームが続行している間は 0 です。

`state`は対戦の状態を表現するもので、さまざまな形式が考えられます。実際、この表現方法が強化学習の性能を左右します。少なくとも、敵としてどのポケモンが場に出ているのかが判断できる必要があります。補助技を適切に使うためには、HP や状態異常の情報も必要です。もっといえば、過去のターンでどんな技が使われたかというような履歴情報も行動決定に影響を与える可能性があります。しかしながら情報が多すぎると各情報の重要性の判断が難しくなり、学習が進みにくくなります。今回は、 表 2.1 に示す要素をベクトルとして並べて与えました。この情報の範囲で、相手に与えるダメージが大きい技はどれか、状態異常にさせる技は有効かどうかがわかります。敵の種類はポケモンの種族数 151 次元のベクトルで表現したほうが正確ですが、パターンが増える分学習すべきパラメータが増加し複雑化するためタイプのみとしました。現実的に初代では 1 つのタイプ（2 タイプの組み合わせ）につき有力なポケモンは 1 種類しかいないため、これで問題ないでしょう。ここでは「こんらん」等の状態変化は含まれていません。将来的に技術開発が進むにつれ、与える情報を増やしていくべきです。

`step()`の実際の実装をリスト 2.4 に示します。引数で与えられた `action`をシミュレータ `self.field`上の技情報に変換し、また対戦相手の技をランダムに選択してシミュレータ上で 1 ターン進めます。バトルが終了した場合は勝者を考慮して報酬 `reward`の計算をします。また、ポケモンが「かたくなる」しか覚えていない場合等は決着がつかないので

▼表 2.1　強化学習における状態表現

| 名称 | 要素数 | 説明 |
|---|---|---|
| 敵タイプ | 15 | 敵のタイプに一致する要素を 1 にする。残りは 0。 |
| HP の割合 | 2 | 現在の HP/最大 HP。味方と敵それぞれ（以下同様）。 |
| 状態異常 | 6 | 状態異常（「正常」も 1 要素とする）に対応する要素を 1 にする。 |
| ランク変化 | 12 | ランク変化（こうげき、ぼうぎょ等の上昇・下降）を表す。 |
| | | -6 から 6 の範囲を 0〜1 の実数に変換。 |
| | | （こうげき・ぼうぎょ・とくしゅ・すばやさ・回避率・命中率） |

一定ターン（64 に設定）で引き分けとして終了することにしました。

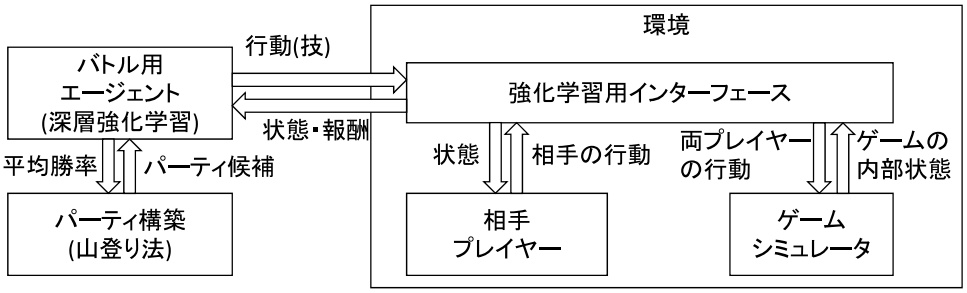
▼リスト 2.4　step 関数の実装

```
def step(self, action: int):
    """
    ターンを進める。
    :param action: 選択する技 (0,1,2,3)、技が N 個ある場合、N 以降を指定した場合は 0 と同
等に扱われる
    :return:
    """
    assert not self.done, "call reset before step"
    assert 0 <= action <= 3
    # 現在選択できる技を取得
    player_possible_actions = self.field.get_legal_actions(0)
    move_idx = action
    # 指定した技が使えるならそれを選択、そうでなければ先頭の技
    # 連続技の最中は選択にかかわらず強制的に技が選ばれる
    player_action = player_possible_actions[0]
    for ppa in player_possible_actions:
        if ppa.action_type is FieldActionType.MOVE and ppa.move_idx == move_idx:
            player_action = ppa
            break
    enemy_action = random.choice(self.field.get_legal_actions(1))
    self.field.actions_begin = [player_action, enemy_action]
    phase = self.field.step()

    reward = 0.0
    if phase is FieldPhase.GAME_END:
        self.done = True
        reward = [1.0, -1.0][self.field.winner]
    else:
        if self.field.turn_number >= PokeEnv.MAX_TURNS:
            # 引き分けで打ち切り
            self.done = True

    return self._make_observation(), reward, self.done, {}
```

## 2.4　山登り法と深層強化学習の組み合わせ

前節までではパーティの構築と行動の選択を独立した手法として説明しました。しかしこれらを組み合わせることで、より強さが増すことが期待できます。ポケモンバトルでは、「10 まんボルト」のような攻撃技（そのターンで直接的に相手にダメージを与える技）と「どくどく」のような補助技（将来的なダメージを増す補助となる技）を織り交ぜながら戦うことが望ましいです。特に、対戦の序盤では補助技を使い、終盤では攻撃技を使うというような使い分けができる必要があります。相手を攻撃技 1 回で倒せる状況で補助技を使うのは望ましくありません。しかし、山登り法でパーティを構築する際はランダムに行動して勝率を測定しているため、使い方の難しい補助技を持ったパーティが生まれづらいと考えられます。実際に、実験では山登り法の結果としてふぶきのみを覚えたパーティが多数生成されました。技が 1 つしかなければ行動に選択肢がありません。このように、ランダムな行動を行う前提で生成されたパーティの行動を深層強化学習で最適化しても、ゲーム全体における強さを最適化したとはいえません。

そこで、山登り法の最中の強さの測定の際に、ランダムに行動するのではなく深層強化学習により適切に行動した場合で測定するほうが適切だと考えられます。以上を踏まえて、システム全体のパイプラインを図 2.2 に示します。




▲図 2.2　システムのパイプライン

疑似コードで表現すると、次のような手順になります。パーティを生成し、その能力を最大限発揮できるように行動選択を最適化した状態で勝率を測定、それをもとに次のパーティを生成するという交互最適化を実現しています。

1. $X$ =パーティをランダム生成
2. $\pi_X = X$ の行動を深層強化学習
3. $R_X = \pi_X$ で行動した際のレーティングを測定
4. $Y_1, Y_2, ..., Y_n = X$ の近傍パーティを $n$ 個生成
5. $\pi_{Y_i} = Y_i$ の行動を深層強化学習
6. $R_{Y_i} = \pi_{Y_i}$ で行動した際のレーティングを測定
7. $R_{max} = R_X, R_{Y_1}, R_{Y_2}, ..., R_{Y_n}$ のうち最大のものを計算
8. $X = R_{max}$ に対応するパーティ
9. 2. に戻って繰り返し

# 第 3 章

# 実験

前章で提案した手法を実験し、その結果を定量的・定性的に評価します。

## 3.1　タイプ相性の実装ミスについて

実験を行った際のシミュレータにバグがあり、水タイプの技が炎タイプに対してダメージ 2 倍とすべきところ、半減に設定されてしまっていました。本書刊行時点での最新コードでは修正済みです。若干炎タイプのポケモンにとって有利な条件ではありますが、実験結果として炎タイプのポケモンは上位の戦略に現れなかったため影響は軽微とみております。仮にメジャーな水タイプを半減できたとしても上位に来られなかった炎タイプの不遇さが際立ちます。

## 3.2　実験 1: 山登り法

山登り法を用いたパーティ構築を行い、ランダムなパーティと比較します。また、山登り法の際に用いる評価関数 $f$ の違いによる結果が変化するか確認します。$f$ として、ランダムに生成したパーティを対戦相手とした場合と、山登り法で別途生成した比較的強いパーティを対戦相手とした場合を比較します。各パーティのバトル中の行動はランダムです。

山登り法ではパーティの近傍を一度に 10 種類生成し、評価関数が最大となるパーティを選択する操作を 30 回反復しました。この値は予備実験により、計算コストと収束のバランスを確認して決定しました。評価関数では、測定対象のパーティを 100 回対戦させてレートを計算しました。1 回の対戦はおよそ 1 ミリ秒かかりました。1 つのパーティの最適化に 30 秒程度かかることとなります。

1 つの実験条件に対し、初期値をランダムに変えた 1000 個のパーティを生成します。

実験条件ごとにグループ A、グループ B のように表記しています。そのグループ内のパーティ全部でレーティング戦を行い、上位 20 パーティをグループの代表とします。実験条件の比較のため、全グループの代表パーティ全部でレーティング戦を行い、各グループのパーティの平均レートを、グループのレートとします。各グループの生成条件は次のようになっています。

- グループ A: ランダム
- グループ B: 山登り法、評価関数の対戦相手はグループ X
- グループ C: 山登り法、評価関数の対戦相手はグループ Y
- 補助グループ（グループ B・C を最適化する際の対戦相手）
  - グループ X: ランダム
  - グループ Y: 山登り法、評価関数の対戦相手はグループ X

グループ A の代表 20 パーティ、グループ B の代表 20 パーティ、グループ C の代表 20 パーティを合わせた総勢 60 パーティで最終的なレーティング戦を行いました。グループの平均レートを 表 3.1 に示します。

▼表 3.1　山登り法による平均レートの比較

| グループ | パーティ構築手法 | 評価関数での対戦相手 | 平均レート |
|---|---|---|---|
| A | ランダム | - | 1311 |
| B | 山登り法 | ランダム | **1621** |
| C | 山登り法 | 山登り法で生成 | 1566 |

グループ A と比べ、グループ B・C のほうが平均レートが高くなっています。山登り法を使用することによってより強いパーティが生成できたと考えられます。一方、グループ C を最適化する際の対戦相手がグループ B の対戦相手より強いため、より洗練されたパーティが生成できることが期待されましたが、それを否定する結果になりました。

各グループの上位 20 パーティに含まれていたポケモンを表 3.2 に示します。グループ A はランダム生成のため、ポケモンの種類というよりは強力な技を覚えていたかどうかに左右されやすいです。グループ B・C を比較すると、グループ C ではルージュラの採用回数が大幅に少ないことがわかります。実は、グループ C の最適化の前の初期値の時点でルージュラが 1 匹しか含まれていませんでした。初期値として 1000 パーティをランダムに生成しますが、ポケモンの種類は 149 種類（ミュウ・ミュウツーは含まないため）からランダムに選択されます。そのため統計的な変動として特定のポケモンが 10 回以上採用される場合もあれば、1 回も採用されないということも起こりえます。最適化の途中でポケモンの種類の変更は行わないため、初期値で強力な種類が入っていなければ最終的な結果の強さが抑えられてしまうという制限が大きく影響しました。

▼表 3.2　ポケモンごとの各グループ上位パーティでの採用回数

| ポケモン | A | B | C |
|---|---|---|---|
| カビゴン | 1 | | |
| ガルーラ | 2 | | |
| ゲンガー | 4 | | |
| ゴース | 1 | | |
| ゴースト | 3 | | |
| サンダー | 2 | | |
| サンダース | | | 1 |
| ジュゴン | 1 | 3 | 4 |
| スターミー | 1 | | |
| スリーパー | | 1 | |
| パルシェン | 1 | | 1 |
| フーディン | | 2 | 3 |
| プテラ | 1 | | |
| フリーザー | 3 | 2 | 3 |
| ラプラス | | 6 | 7 |
| ルージュラ | | 6 | 1 |

各グループの上位 20 パーティの具体的な内容とレートを表 3.3 表 3.4 表 3.5 に示します。

すがすがしいほどのふぶき無双となりました。全体としては、ルージュラ・ラプラスが最強クラスの性能のようです。ふぶきの効果が 27% で相手を実質即死させるというルールに設定しているため、何も考えずにこれを連発するのは妥当でしょう。氷タイプのポケモン（ラプラス・ルージュラ・フリーザー・ジュゴン等）は凍らないので、自分がふぶきを使えるだけでなく相手のふぶきへの対抗手段としても有力です。この実験ではバトル中の技の選択はランダムであるため、技の枠 4 つを使い切らず不要な（または使いどころが難しい）技が消えた状態が最適という結果になっています。上位はほとんど氷タイプのポケモンで占められていますが、最強だったのは電気タイプのサンダースでした。電気技に耐性のあるポケモンが軒並みふぶきに弱いため、上位にまったく現れなかったことが要因と考えられます。なお、まったく同じ構成のパーティでも、レートが 100 程度開く場合があるようです。技の命中や対戦相手の選択における乱数により差が生じています。

ちなみに、各グループ内での最弱パーティを表 3.6 表 3.7 に示します。グループ A では技がランダムなので、「じばく」「だいばくはつ」を覚えている場合が最弱でした。パーティのポケモンが 1 匹しかいないときに使うと反則負けなので当然の帰結です。グループ B では最適化を行っているためこれらの技は忘れられます。この場合最弱となるのはまともな技を覚えないトランセル・ビードル等のポケモンを採用した場合でした。初期値から

▼表 3.3　グループ A の代表パーティのレートおよび構成

| レート | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|
| 1650 | ジュゴン | かげぶんしん | ふぶき | どくどく | ねむる |
| 1556 | サンダー | かみなり | 10 まんボルト | そらをとぶ | でんじは |
| 1487 | パルシェン | とっしん | かげぶんしん | とげキャノン | ふぶき |
| 1377 | フリーザー | こうそくいどう | れいとうビーム | ふぶき | はかいこうせん |
| 1357 | フリーザー | つつく | ゴッドバード | ふぶき | れいとうビーム |
| 1344 | ゴースト | サイコキネシス | さいみんじゅつ | メガドレイン | ゆめくい |
| 1323 | ゴースト | さいみんじゅつ | したでなめる | かみなり | ゆめくい |
| 1321 | ゲンガー | メガドレイン | ちきゅうなげ | はかいこうせん | かみなり |
| 1312 | フリーザー | すてみタックル | そらをとぶ | れいとうビーム | かげぶんしん |
| 1297 | サンダー | 10 まんボルト | はかいこうせん | ふきとばし | ゴッドバード |
| 1294 | ゲンガー | したでなめる | とっしん | サイコウェーブ | かみなり |
| 1294 | カビゴン | のしかかり | はかいこうせん | とっしん | メガトンキック |
| 1277 | スターミー | れいとうビーム | サイコキネシス | バブルこうせん | どくどく |
| 1274 | ゴースト | したでなめる | ナイトヘッド | メガドレイン | 10 まんボルト |
| 1225 | ゴース | メガドレイン | サイコキネシス | サイコウェーブ | かみなり |
| 1215 | ゲンガー | 10 まんボルト | かいりき | じごくぐるま | メガドレイン |
| 1189 | ゲンガー | かげぶんしん | サイコウェーブ | かいりき | のしかかり |
| 1161 | ガルーラ | メガトンキック | はかいこうせん | 10 まんボルト | れいとうビーム |
| 1141 | プテラ | つばさでうつ | だいもんじ | りゅうのいかり | そらをとぶ |
| 1133 | ガルーラ | じわれ | かいりき | メガトンキック | いわなだれ |

ポケモンの変更ができないという手法の制限による結果です。

結論として、山登り法でパーティ構成の強化が可能であることがわかりました。一方で、初期値への依存性が大きく出る点の克服が今後の課題だと分かりました。バトルのルールとして、ふぶきの凍る確率を 30% に設定したのはやはり強すぎるようです。最適化の効果を確認しやすい一方、戦略の幅が狭まってしまった懸念があります。

## 3.3　実験 2: 強化学習

この実験では、パーティ構成は固定して、バトル中の技の選択を強化学習により選択した場合の効果を確認します。

パーティ構成には、実験 1 で生成したグループ B の上位 20 パーティを用いました。そして、強化学習の際の対戦相手となるパーティグループを 2 種類用意し、それによる差異も確認します。

- グループ B: 実験 1 で生成したパーティのまま、ランダム行動

▼表 3.4　グループ B の代表パーティのレートおよび構成

| レート | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|
| 1761 | ルージュラ | サイコキネシス | あくまのキッス | | |
| 1754 | ルージュラ | サイコキネシス | あくまのキッス | | |
| 1748 | ルージュラ | サイコキネシス | | | |
| 1712 | ラプラス | ふぶき | 10 まんボルト | | |
| 1711 | ルージュラ | サイコキネシス | | | |
| 1708 | ラプラス | ふぶき | 10 まんボルト | | |
| 1693 | ラプラス | ふぶき | サイコキネシス | 10 まんボルト | |
| 1665 | フーディン | サイコキネシス | | | |
| 1656 | スリーパー | サイコキネシス | | | |
| 1644 | フーディン | サイコキネシス | | | |
| 1630 | ラプラス | ふぶき | | | |
| 1616 | ラプラス | ふぶき | サイコキネシス | | |
| 1603 | ルージュラ | サイコキネシス | | | |
| 1603 | ラプラス | ふぶき | | | |
| 1571 | ルージュラ | ふぶき | あくまのキッス | | |
| 1528 | ジュゴン | ふぶき | | | |
| 1512 | ジュゴン | ふぶき | | | |
| 1484 | ジュゴン | ふぶき | | | |
| 1418 | フリーザー | ふぶき | | | |
| 1405 | フリーザー | ふぶき | | | |

- グループ D: グループ B をパーティ構成変化させずに、グループ X（ランダム生成）を相手に強化学習
- グループ D’: グループ D のパーティでランダム行動（グループ B と完全に同じ）
- グループ E: グループ B をパーティ構成変化させずに、グループ Y（山登り法で生成）を相手に強化学習
- グループ E’: グループ E のパーティでランダム行動（グループ B と完全に同じ）

深層強化学習アルゴリズムには DQN を用い、DNN の構造には隠れ層 2 層、各 32 次元の全結合ネットワークを用いました。Optimizer は Adam、探索には Epsilon Greedy を用いています。学習は 10,000 ステップ行いました。1 ステップは 1 ターンに相当し、バトルが終了するたび対戦相手がランダムに変わります。深層学習では一般に CPU ではなく GPU が用いられますが、今回は使用していません。GPU は画像認識で用いられるような複雑なモデルでは大きな効果を発揮しますが、今回のような小さなモデルだと制御コストが支配的になり、高速化に寄与しません。

各グループ 20 パーティ、総勢 100 パーティ間でレーティング戦を行い、グループ内の

▼表 3.5　グループ C の代表パーティのレートおよび構成

| レート | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|
| 1816 | サンダース | 10 まんボルト | | | |
| 1716 | フーディン | サイコキネシス | | | |
| 1702 | ラプラス | ふぶき | うたう | サイコキネシス | |
| 1696 | ラプラス | ふぶき | うたう | | |
| 1620 | ラプラス | ふぶき | | | |
| 1605 | ラプラス | ふぶき | | | |
| 1604 | ラプラス | ふぶき | | | |
| 1601 | ラプラス | ふぶき | | | |
| 1573 | フーディン | サイコキネシス | | | |
| 1562 | ルージュラ | あくまのキッス | ふぶき | ちきゅうなげ | |
| 1554 | ジュゴン | ふぶき | | | |
| 1536 | ラプラス | ふぶき | | | |
| 1530 | フーディン | サイコキネシス | | | |
| 1530 | ジュゴン | ふぶき | | | |
| 1483 | ジュゴン | ふぶき | | | |
| 1476 | フリーザー | ふぶき | | | |
| 1471 | ジュゴン | ふぶき | | | |
| 1458 | フリーザー | ふぶき | | | |
| 1423 | フリーザー | ふぶき | | | |
| 1363 | パルシェン | ふぶき | | | |

▼表 3.6　グループ A 最弱パーティの構成

| ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|
| マルマイン | フラッシュ | ソニックブーム | でんじは | だいばくはつ |
| ゴースト | かげぶんしん | ゆめくい | したでなめる | じばく |
| ゴースト | じばく | あやしいひかり | かげぶんしん | ねむる |
| ドガース | かげぶんしん | じばく | かみなり | たいあたり |
| マルマイン | たいあたり | じばく | はかいこうせん | かげぶんしん |

平均レートを計算しました。グループ D’・E’は実装の都合で存在しています。しかしながら、表 3.4 で示されるように、20 パーティ中 7 パーティしか複数の技を覚えていません。覚えている技が 1 つだけだと学習の余地がないため、複数の技を覚えている 7 パーティの中で平均レートを計算しました。

結果を表 3.8 に示します。平均レートがすべて 1500 より高いのは、ふぶきのみ覚えたフリーザー等の下位パーティが統計対象外にいるためです。グループ B・D はほとんど差異がなく、グループ E は若干レートが高いという結果になりました。グループ D では対

▼表 3.7　グループ B 最弱パーティの構成

| ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|
| トランセル | かたくなる | | | |
| キャタピー | たいあたり | いとをはく | | |
| キャタピー | たいあたり | | | |
| トランセル | かたくなる | たいあたり | いとをはく | |
| ビードル | どくばり | いとをはく | | |

▼表 3.8　強化学習による平均レートの比較

| グループ | 行動選択 | 強化学習時の相手 | 平均レート |
|---|---|---|---|
| B | ランダム | - | 1626 |
| D | 強化学習 | ランダム | 1640 |
| E | 強化学習 | 山登り法で生成 | **1702** |

戦相手がランダムに生成されたもののため、状況に応じた技の選択を行わずとも勝率が十分高く、学習が進まなかったのではないかと考えられます。グループ E では対戦相手が強く、技の選択が下手だと負けてしまうため学習がうまく進んだものと考えられます。

レート最大だったのは、グループ E のルージュラ（技＝サイコキネシス・あくまのキッス）で 1975 でした。同パーティでランダムに行動した場合は 1687 でした。対戦記録を見ると、相手が眠っていなければあくまのキッスで眠らせて、眠っていればサイコキネシスで攻撃という理想的な戦略を取れていました。強化学習により補助技の活用法を学習できたと考えられます。

一方で、強化学習を行ってもレートにほとんど変化がなかったパーティもあります。ラプラス（技＝ふぶき・10 まんボルト）は相手のタイプによって技を選ぶことで強くなりそうですが、うまくいきませんでした。補助技と違いどちらの技を選んでもほとんどの場合勝てるので、戦略の優劣が付きにくかったのかもしれません。学習手法については改善の余地があります。

## 3.4　実験 3: 交互最適化

山登り法と深層強化学習の交互最適化により、山登り法の結果のパーティの行動を深層強化学習する場合との違いを比較します。

交互最適化は、山登り法における評価関数の内部で強化学習を行うため非常に高コストとなり、1 パーティあたり 3 時間程度計算時間がかかります。ランダムなパーティを初期値とした場合、トランセル等弱い種族の検討も行うこととなり時間が無駄になります。そのため初期値としてランダムなパーティではなく山登り法で得られたパーティを用いるこ

ととしました。グループ Z がその初期値となります。

- グループ B: 実験 1 で生成したパーティのまま、ランダム行動
- グループ E: グループ B をパーティ構成変化させずに、グループ Y（山登り法で生成）を相手に強化学習
- グループ F: グループ Z を交互最適化（強化学習の相手はグループ Y）
- グループ G: グループ F と同じパーティをランダム戦略でテスト
- グループ H: グループ Z をパーティ構成変化させずに、グループ Y を相手に強化学習（グループ E と同等）
- グループ Z: グループ B と同手法で生成

▼表 3.9　交互最適化による平均レートの比較

| グループ | 行動選択 | パーティ最適化 | 平均レート |
|---|---|---|---|
| B | ランダム | 山登り法 | 1519 |
| E | 強化学習 | 山登り法 | 1548 |
| F | 強化学習 | 山登り法（交互最適化） | **1610** |
| G | ランダム | 山登り法（交互最適化） | 1406 |
| H | 強化学習 | 山登り法 | 1468 |
| Z | ランダム | 山登り法 | 1445 |

結果を表 3.9 に示します。提案手法である交互最適化の平均レートが最大という結果になりました。逆にグループ G は平均レートが最低です。これは、ランダムに使うと効果がない補助技がパーティに含まれているためと考えられます。各グループのパーティおよびレートを表 3.10 表 3.11 表 3.12 に示します。

▼表 3.10　グループ E・B の各パーティのレートおよび構成

| E | B | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|---|
| 1901 | 1729 | ルージュラ | サイコキネシス | あくまのキッス | | |
| 1896 | 1860 | ルージュラ | サイコキネシス | あくまのキッス | | |
| 1862 | 1584 | ルージュラ | ふぶき | あくまのキッス | | |
| 1710 | 1545 | ラプラス | ふぶき | 10 まんボルト | | |
| 1660 | 1641 | ルージュラ | サイコキネシス | | | |
| 1599 | 1437 | フーディン | サイコキネシス | | | |
| 1598 | 1568 | ラプラス | ふぶき | サイコキネシス | 10 まんボルト | |
| 1595 | 1627 | ルージュラ | サイコキネシス | | | |
| 1584 | 1530 | ルージュラ | サイコキネシス | | | |
| 1558 | 1540 | ラプラス | ふぶき | 10 まんボルト | | |
| 1514 | 1501 | ラプラス | ふぶき | | | |
| 1503 | 1541 | フーディン | サイコキネシス | | | |
| 1474 | 1540 | ラプラス | ふぶき | サイコキネシス | | |
| 1407 | 1464 | ラプラス | ふぶき | | | |
| 1407 | 1433 | ジュゴン | ふぶき | | | |
| 1404 | 1350 | ジュゴン | ふぶき | | | |
| 1392 | 1425 | ジュゴン | ふぶき | | | |
| 1314 | 1381 | フリーザー | ふぶき | | | |
| 1311 | 1300 | フリーザー | ふぶき | | | |
| 1276 | 1391 | スリーパー | サイコキネシス | | | |

▼表 3.11　グループ F・G の各パーティのレートおよび構成

| F | G | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|---|
| 1935 | 1338 | ルージュラ | ちきゅうなげ | あくまのキッス | ロケットずつき | じごくぐるま |
| 1876 | 1442 | ルージュラ | ちきゅうなげ | あくまのキッス | ねむる | |
| 1872 | 1472 | ルージュラ | あくまのキッス | ちきゅうなげ | れいとうビーム | ねむる |
| 1856 | 1384 | ルージュラ | ふぶき | あくまのキッス | はたく | |
| 1734 | 1516 | ルージュラ | ふぶき | はかいこうせん | あくまのキッス | |
| 1713 | 1366 | ジュゴン | ふぶき | ねむる | なみのり | なきごえ |
| 1690 | 1500 | ラプラス | ふぶき | はかいこうせん | | |
| 1636 | 1149 | フーディン | サイコキネシス | フラッシュ | | |
| 1612 | 1569 | ジュゴン | ふぶき | どくどく | すてみタックル | |
| 1584 | 1482 | ラプラス | ふぶき | はかいこうせん | | |
| 1547 | 1294 | フーディン | サイコキネシス | かげぶんしん | メガトンキック | |
| 1527 | 1528 | ジュゴン | ふぶき | ずつき | かげぶんしん | |
| 1514 | 1473 | パルシェン | ふぶき | はかいこうせん | | |
| 1510 | 1438 | ラプラス | ふぶき | うたう | | |
| 1510 | 1403 | フーディン | サイコキネシス | フラッシュ | | |
| 1507 | 1326 | フーディン | サイコキネシス | どくどく | じごくぐるま | |
| 1503 | 1412 | フーディン | サイコキネシス | トライアタック | | |
| 1462 | 1312 | ラッキー | ふぶき | タマゴばくだん | 10 まんボルト | |
| 1331 | 1457 | パルシェン | ふぶき | かげぶんしん | バブルこうせん | |
| 1292 | 1262 | フリーザー | ふぶき | はかいこうせん | | |

▼表 3.12　グループ H・Z の各パーティのレートおよび構成

| H | Z | ポケモン | 技 1 | 技 2 | 技 3 | 技 4 |
|---|---|---|---|---|---|---|
| 1789 | 1469 | ラプラス | ふぶき | うたう | | |
| 1730 | 1536 | ルージュラ | ふぶき | サイコキネシス | | |
| 1601 | 1580 | ルージュラ | サイコキネシス | | | |
| 1568 | 1571 | ルージュラ | サイコキネシス | | | |
| 1565 | 1520 | フーディン | サイコキネシス | | | |
| 1558 | 1453 | フーディン | サイコキネシス | | | |
| 1553 | 1610 | フーディン | サイコキネシス | | | |
| 1538 | 1558 | フーディン | サイコキネシス | | | |
| 1520 | 1431 | ラプラス | ふぶき | | | |
| 1508 | 1517 | フーディン | サイコキネシス | | | |
| 1442 | 1396 | ジュゴン | ふぶき | | | |
| 1439 | 1475 | ラプラス | ふぶき | | | |
| 1417 | 1447 | ジュゴン | ふぶき | | | |
| 1390 | 1301 | ルージュラ | ふぶき | | | |
| 1385 | 1461 | ジュゴン | ふぶき | | | |
| 1360 | 1393 | パルシェン | ふぶき | | | |
| 1300 | 1327 | フリーザー | ふぶき | | | |
| 1284 | 1324 | ルージュラ | ふぶき | | | |
| 1234 | 1388 | パルシェン | ふぶき | | | |
| 1173 | 1148 | ラッキー | ふぶき | | | |

最強のパーティはルージュラ（技＝ちきゅうなげ・あくまのキッス・ロケットずつき・じごくぐるま）でした。ルージュラ同士の対面だと、ダメージ計算ツールによればサイコキネシスでは相手を倒すのに 6 発以上、ふぶきでは 5 発以上かかりますが、ちきゅうなげなら 4 発のためグループ E の技構成より有利といえます。

対戦記録の例を図 3.1 に掲載します。あくまのキッスで眠らせてからちきゅうなげかじごくぐるま連打という戦略をとっています。ルージュラは素早さが高く、また初代のルールでは目覚めたターンには行動できないため、この戦法で相手を完封できる可能性が高く強力だといえます。ロケットずつきは使われていません。技選択の学習において、技 3 つで十分であれば 4 つ目はランダムな技が入ったまま使われないという事象が起きるようです。相手を眠らせることが非常に強いため、攻撃技がそこまで強くなくても勝てるようです。このパーティは全勝ではなく、フーディンとの戦いであくまのキッスを 2 回外し、相手のサイコキネシスが 2 回急所に当たり負けるというバトルがありました。攻撃技としてふぶきやサイコキネシスを採用すればもっと強くなることが容易に想像できるため、手法にまだまだ改善の余地がありそうです。

また、相手によって戦略を使い分けるパーティもありました。ジュゴン（技＝ふぶき・どくどく・すてみタックル）は、ラプラス相手では最初にどくどくを使った後すてみタックルを使用、その他の相手では常にふぶきを使用という戦略でした。相手のタイプ情報を参照して行動を変える戦略も実現可能であることがわかりました。

以上のように、提案手法によって比較的よいパーティ構成および技選択が行えることが確認できました。戦略の面で判明したこととしては、パーティがポケモン 1 匹だけの場合は、やはり氷タイプが最強で、その中で同じ氷タイプへの対策ができるか否かという点が強さを決めるということがわかりました。

| パーティ1 | | パーティ2 |
|---|---|---|
| ターン1 | | |
| ルージュラを繰り出した | | ラプラスを繰り出した |
| ルージュラのあくまのキッス | → | ラプラスは眠った |
| | ← | 眠っている |
| ターン2 | | |
| ルージュラ HP188 | | ラプラス HP259 |
| ルージュラのちきゅうなげ | → | ラプラス HP259→204 |
| | ← | 眠っている |
| ターン3 | | |
| ルージュラ HP188 | | ラプラス HP204 |
| ルージュラのすてみタックル<br>反動ダメージ HP188→174 | → | ラプラス HP204→147 |
| | | 目を覚ました |
| ターン4 | | |
| ルージュラ HP174 | | ラプラス HP147 |
| ルージュラのあくまのキッス | → | ラプラスは眠った |
| | ← | 眠っている |
| ターン5 | | |
| ルージュラ HP174 | | ラプラス HP147 |
| ルージュラのすてみタックル<br>反動ダメージ HP174→161 | → | ラプラス HP147→93 |
| | ← | 眠っている |
| ターン6 | | |
| ルージュラ HP161 | | ラプラス HP93 |
| ルージュラのすてみタックル | → | 外れた |
| | ← | 眠っている |
| ターン7 | | |
| ルージュラ HP161 | | ラプラス HP93 |
| ルージュラのすてみタックル<br>反動ダメージ HP161→147 | → | ラプラス HP93→34 |
| | ← | 眠っている |
| ターン8 | | |
| ルージュラ HP147 | | ラプラス HP34 |
| ルージュラのすてみタックル<br>反動ダメージ HP147→139 | → | ラプラス HP34→0 |
| | | ラプラスは倒れた |
| 試合終了 勝者 パーティ1 | | |

▲図 3.1　最強パーティの対戦記録

# 第 4 章

# まとめ

本書では、テレビゲーム「ポケットモンスター」におけるポケモンバトルの戦略を、ゲーム AI の観点で研究しました。どんな技が強いか、どんな立ち回りをすべきかということを人が教えずとも自動的に発見できるようなシステムを作成することを目的とし、どのような技術が応用可能か検討しました。ルールはもっとも単純な、赤・緑バージョンにおいてポケモン 1 匹のみを使用する条件で検討しました。パーティの構築には離散最適化の一種である山登り法を、バトル中の行動の選択には強化学習の一種である深層強化学習（DQN）をベースとした手法を提案しました。そして、これらの手法を交互に繰り返すことでより強力な戦略を生み出せることを実験で確認しました。

実験で構築された戦略のうち最強のものは、ポケモンとしてルージュラ（技＝ちきゅうなげ・あくまのキッス・ロケットずつき・じごくぐるま）を採用し、あくまのキッスで眠らせてからちきゅうなげかじごくぐるま連打という戦略でした。複数の技を組み合わせることができており多少賢い戦略ではありますが、まだまだ強化の余地がありそうです。

今後の課題として、行動選択において攻撃技のうちより適切なものを選択できるよう、学習手法を改善することが望ましいです。また、パーティ構成を複数体のポケモンに拡張し、より本来のポケモンバトルに近い条件に対応できる手法の開発を行いたいと考えています。

# あとがき

筆者がポケモンと出会って約 20 年が経ちました。緑バージョンを買った当時小学生だった自分が、それを題材にこんな研究をするとはまったく予想できませんでした。

プロジェクトの構想自体は 2012 年頃からあったのですが、強化学習などの手法に対する自分の理解、ソフトウェアライブラリの未成熟によりなかなか実現しませんでした。2017 年に ChainerRL という扱いやすい強化学習ライブラリが公開されたのを機にプロジェクトを再始動し、今度こそ一定の成果を得ることに成功しました。

大学に入って上京し、コミケに参加して技術系同人誌という存在を知り、いつか自分も出したいと思うようになりました。2017 年末に「準備編」と題して PDF を公開しましたが、今回ついに念願のイベント参加・紙媒体での発行がかないました。

謝辞。ポケモンという素晴らしいゲームを開発してくださったゲームフリークの皆様。毎バージョン楽しませていただいています。Switch での新作も期待しています。「技術書をかこう！」という本を配布された TechBooster の皆様。書籍テンプレート `https://github.com/TechBooster/ReVIEW-Template` を利用させていただいています。また、校正には noe さんに協力していただきました。

ポケモンバトルの究極の戦略を見るために、これからも研究を続けたいと思います。文章力の低さ、出版に関する知識不足等至らぬ点もあるかと思いますが、お読みいただきありがとうございました。

本書の電子版を https://select766.booth.pm/items/1023281 からダウンロードできます。解凍パスワード：bS6V5vjd2X

# ＰｏｋéＡＩ ～人工知能の考えた最強のポケモン対戦戦略～（１）

2018 年 10 月 8 日　技術書典５ v1.1.0
2019 年 4 月 14 日　技術書典６ v1.1.1 （誤植訂正）
著　者　select766 (select766@outlook.jp)
発行所　ヤマブキ計算所
印刷所　ねこのしっぽ

# ヤマブキ計算所