PokéAI

～人工知能の考えた最強のポケモン対戦戦略～

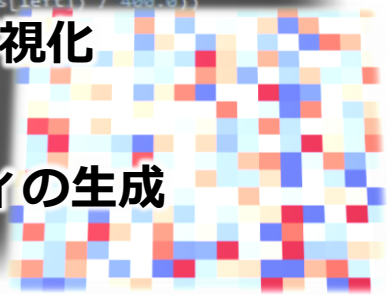
［#3：金銀導入編］

▷オープンソースシミュレータを用いた AI 開発

▷パーティ構成最適化における持ち物対応

▷強化学習における思考パターンの可視化

▷付録：「リトルカップ」対応パーティの生成



@select766 / ヤマブキ計算所

まえがき

PokéAI（ポケエーアイ）の世界へようこそ。PokéAI 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）に考えさせる研究プロジェクトです。

本書は 2019 年 4 月発行の第 2 巻の続編となります。第 1・2 巻では、ポケモン赤緑（初代）のルールにおけるパーティ生成・行動選択の最適化を試みました。最低限、有効だと思える戦略を確認することができたものの、大いに改良の余地を残す結果でした。初代では、場をかき乱すような補助技のレパートリーに欠けるため、単純に威力の高い技でごり押す戦略が有効であり、AI の出力した戦略もそのような傾向が強くなっていました。第 3 巻では、ゲームバランスが改良され知的な戦略の有効性がより高まった環境として、ポケモン金銀（第 2 世代）のルール上で AI 開発を行います。本巻ではルール・シミュレータが大きく変更となったことに対応するソフトウェア実装が主題となります。対戦形式のゲームに対する AI 開発の資料は、チェス・将棋・囲碁を除けば多くありません。本書で示す技法がポケモン以外のゲームに有効かどうかはわかりませんが、泥臭い処理も含めて開発の雰囲気を感じていただければと思います。

本書は機械学習の知識を前提としませんが、紙面の都合上用いる技術やそれが内包する問題点をすべて解説することはできません。その場合でも最低限のアイデアが伝わるような図を用いるよう心がけています。また、過去の巻がなくても理解できるようプロジェクト概要・ポケモンバトルのルール説明等を再掲しており重複となりますがご了承ください。

目次

第 1 章

イントロダクション

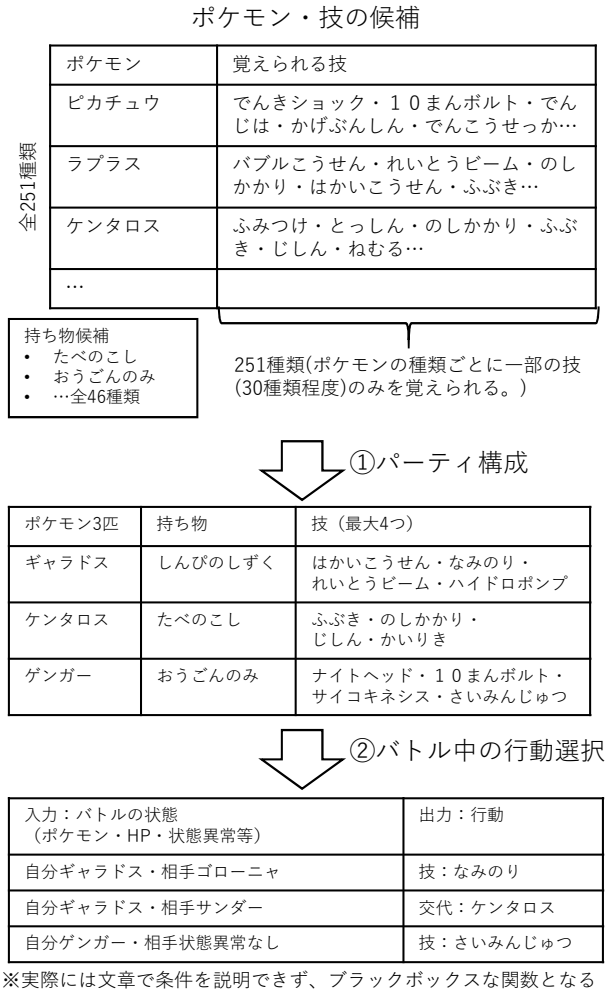
PokéAI は、ポケモンバトルの戦略を人工知能（AI）に考えさせるプロジェクトです。「ふぶきが強い」というような人間の知識を極力使わずに、ポケモンバトルのルールから AI が自律的に強力な戦略を発見することを期待します。2019 年現在の人工知能技術ではこの課題の解き方は自明ではなく、他のゲームで培われた技術をもとに、ポケモンバトルに必要な技術開発を行っていく必要があります。そして、実験結果として得られた AI の戦略を鑑賞することが大きな楽しみになります。特に、ポケモンバトルは「パーティの生成」と「バトル中の行動選択」という 2 つの段階の組み合わせが必要となる点が特徴的です。

ポケモンのゲームソフト内で登場する敵トレーナーの行動を決定する機能も一種の AI です。たとえば、ジムリーダーなどの強いトレーナーは、こちらのポケモンに効果が抜群となるタイプの技を選択するようにプログラミングされています。しかし、このような AI はプログラマーが条件と行動の組み合わせを逐一設計したもの（ルールベース）であり、プログラマーが思いつかなかった戦略を発揮することはできません。本書では、条件と行動を結びつけるパラメータを人が決めるのではなく、バトルの勝敗から自動的に決定する機構を作ることで、人が思いつかなかった戦略さえも取ることが可能な AI を開発していきます。

本書では、パーティの構成と、バトル中の行動選択の両方を AI に最適化させます。全体の流れを図 1.1 に示します。筆者の知る限り、従来研究[*1]ではパーティの構成はランダムに決める、またはポケモンのタイプ補完だけ考慮するというものしか見つからないのですが、やはりこの 2 つの段階両方をうまくかみ合わせることが、ランダムにカードが配ら

[*1] Showdown AI Competition (2017): http://game.engineering.nyu.edu/showdown-ai-competition/
ポケモンバトルの AI を作ってみたよ: http://shingaryu.hatenablog.com/entry/2016/02/03/200000

れるトランプゲームや麻雀とポケモンバトルの大きな違いだと筆者は考えているため、両方を AI で検討することを目的としています。



▲図 1.1　ポケモンバトル AI の 2 つの課題。

既刊となる第 1・2 巻では、1996 年発売のポケモン赤緑（初代）のポケモンバトルのルールに準拠して AI を開発しました。第 3 巻となる本巻の目的は、1999 年発売のポケモン金銀（第 2 世代）のルールに対応した AI の開発の第一歩を踏み出すことです。初代ルールはゲームバランスに難があり、活躍できるポケモン・技が狭いという問題点がありました。特に複数の技を絡めた戦法があまり有効ではなく、AI の技術を進歩させても興

味深い戦法を観察できる可能性が低いという懸念がありました。金銀ルールではこれらの問題が大きく改良されました。たとえば「にほんばれ」で炎タイプの技のダメージを 1.5 倍にしたうえで「だいもんじ」で攻撃したり、「ねむる」で体力を回復しながら眠り状態でも技を出せる「ねごと」で攻撃するなど、技を組み合わせて相乗効果を得る戦いが当たり前になりました。技術開発には長い道のりが予想されますが、AI が賢くなればこれらの戦法を自動的に発見したり、また未知の戦法にも到達できるかもしれません。このような目標に向かうために、本巻では主にソフトウェアシステムの構築に取り組みました。初代ルールに対しては、これに準拠したポケモンバトルシミュレータを独自に開発しました。しかしこれにはかなりの時間を要しており、より複雑な金銀ルールへの対応を行うことにコストを割くと本命の AI 開発が進まなくなってしまうことが予想されました。そのため、オープンソースで提供されている既存の非公式ポケモンバトルシミュレータである Pokémon-Showdown を利用する方針へと転換しました。Pokémon-Showdown は人間同士がオンラインで対戦して遊ぶために作られたものであり、これを独自の AI と接続するためのインターフェースの実装が必要です。「2.1 オープンソースシミュレータの利用」では、このインターフェースの実装について解説します。

次に、金銀ルールを採用したことによるアルゴリズム面の変更を行います。ルールの大きな変更点として、ポケモンに道具を持たせることができるようになりました。ポケモンが持っている道具を持ち物と呼びます。1 匹のポケモンには 1 つの道具を持たせることができ、バトル中に特定の条件を満たした時に自動的に機能が発動します。たとえば、「しんぴのしずく」を持ったポケモンが水タイプの攻撃技を使うと、威力が 1.1 倍になります。どのポケモンにどの持ち物を持たせるかという事項を AI においても考慮するように拡張します。パーティの生成は「2.2 パーティの生成」にて解説します。ルールのもうひとつの変更点として、「場」に影響する効果が導入されました。主たる効果は天候です。天候変化を発生させたポケモンが倒されたり交代したりしても、その効果が継続します。そのため、あるポケモンが発動させた効果を後続のポケモンが引き継いで戦うような戦術の可能性が大きく広がりました。強化学習における入力として天候を追加しますが、本巻の実験結果ではそれを活用するには至っていません。バトル中の行動選択は「2.3 Actor-Critic 型強化学習による行動方策の学習」にて解説します。

1.1 ポケモンバトルのシステム

第 2 巻とほぼ同様となりますが、ポケモンバトルの基本的なシステム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、金銀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）、先に手持ちの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。

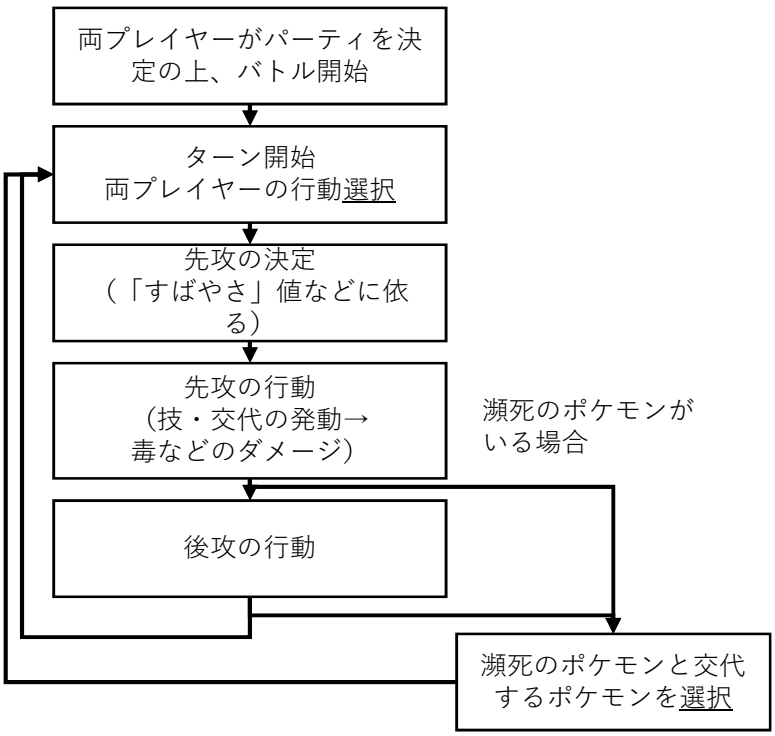
1 人のプレイヤーは 1〜6 匹（本書のルールでは 3 匹）のポケモンを持ちます。このポケモンの組をパーティと呼びます。ポケモンは 1 匹につき最大 4 種類の技を覚えることができます。ポケモンに覚えさせられる技の候補は 4 種類より多く、どの技を覚えさせておくかは戦略に依存する要素です。また、ポケモンの種類により覚えられる技は異なります。たとえば、氷タイプ[*2]のポケモンであるフリーザーはふぶきを覚えられますが、かえんほうしゃは覚えられません。また、金銀ルールでは新たにポケモンに道具を持たせることが可能となります。ポケモンに持たせる道具を特に「持ち物」と呼ぶこともあります。持ち物には、水タイプの技の威力を 1.1 倍にする「しんぴのしずく」や、毒状態になった際に自動的に回復する「どくけしのみ」など約 40 種類があります。持ち物はプレイヤーがコマンドを選択して発動させるのではなく、発動条件が満たされると自動的に機能が発揮されます。また、相手の持ち物を奪って自分のものにする「どろぼう」という技など、直接的に持ち物に干渉するような技も用意されています。ポケモン 1 匹あたり最大 1 つの持ち物を持たせることが可能です。技と違い、ポケモンと持ち物の組み合わせ方に制約はありません。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、すなわち対戦相手の情報を知る前に行う作業です。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。また、同じ種類のポケモンであっても能力値に違いがありますが、最大値に固定します。すなわちいわゆる個体値・努力値をすべて最大値に設定します。

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代するかの 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代は、瀕死になっていない任意のポケモンを選択して交代できます。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

[*2] 正式なゲーム用語はひらがなですが、読みやすさやスペースの都合で漢字を用いる場合があります。

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。「選択」と書かれた部分を AI が行う。

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。なお、補助技にもタイプが設定されていますが、ほとんどの場合影響はありません。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。

1.2 本書で扱うポケモンバトルのルール設定について

ポケモン金銀の標準的なルールに従いますが、AI で扱うにあたり特別な考慮の必要な技などについて一部制約を設けています。これはパーティ生成時の選択肢から除外することで実現します。またシミュレータは非公式のものであり、実機の挙動とは異なる可能性があります。

プレイヤー 2 人のシングルバトルで、プレイヤー 1 人は 3 匹のポケモンを所持します。バトル前の見せ合いはありません。ポケモンのレベルは 55、50、50[*3]とし、先頭のポケモンがレベル 55 とは限りません。3 匹のポケモンはすべて異なる種類とし、また持ち物も異なる種類とします。ポケモンの個体値（個体ごとに異なる強さパラメータ）、努力値（育成によって増加する強さパラメータ）はすべて最大値に固定します。ポケモン金銀ではポケモンの性別はオスとメス両方が存在する種族では攻撃の個体値に依存して決まり、この場合はオス固定となります。ポケモンが覚える技については、そのレベルに応じた制約に準拠します（例：レベル 50 のファイヤーはにらみつけるを覚えられない）。クリスタルバージョンでのみ覚えられる技（10 まんボルトなどの教え技）やイベント配布ポケモンが特別に覚えている技[*4]を含みます。

技の挙動が対戦相手、控えのポケモン、バトル中の過去の行動などに依存するものは、適切な運用のために入力すべき情報量が増加し、各情報の重要度を判断させることが難しくなるため除外しています。

使用禁止技：バトンタッチ・ふくろだたき・かなしばり・めざめるパワー・ゆびをふる・ものまね・オウムがえし・スケッチ・うらみ・どろぼう・へんしん

技の制約および慣習により、次のポケモンはパーティに加えません。

使用禁止ポケモン：メタモン・ミュウツー・ミュウ・ドーブル・ルギア・ホウオウ・セレビィ

[*3] 現代ではレベルは 50 統一が普通なのですが、当時はレベル配分が戦略の要素になっていました。本来のルールでは、各ポケモンのレベルが 50 から 55 で、ポケモン 3 匹の合計が 155 という制約でしたので、52、52、51 というようなレベル配分も可能です。ポケモンに性格がない当時、能力値配分による特徴付けを可能とする有力な方法であり、またレベル 55 で進化するカイリュー・バンギラスをパーティに入れるかどうかの選択もキーとなります。

[*4] 使用したデータベースに含まれていた意外な組み合わせとして、海外限定の配布ポケモンで、あくまのキッスを覚えたニドラン♀という事例がありました。

第 2 章

金銀ルールに対応したポケモンバトル AI の実装

2.1　オープンソースシミュレータの利用

本巻からは、ポケモンバトルのシミュレータとして自作ではなく第三者が開発したオープンソースのシミュレータを用いることとしました。調査の結果、AI への活用の先行事例がある Pokémon-Showdown を用いることとしました。

2.1.1　Pokémon-Showdown の概要

Pokémon-Showdown [*1] は、インターネット上でポケモンバトルを楽しめる非公式のサービスです。ポケモンの育成はなく、プレイヤーが入力したパーティ構成でバトルを行うことができます。

2.1.2　シミュレータの構造

Pokémon-Showdown はシミュレータのほか、ユーザーが直接操作する GUI やユーザーの管理機能などもすべて含んでいます。AI 開発に必要なのはシミュレータ単体ですので、その部分を取り出す手段を調べました。

Pokémon-Showdown のソースコードは GitHub 上で公開されています。機能別に複数のリポジトリに分かれており、シミュレータ部分は[*2]に含まれています。シミュレータのソースコードは主に TypeScript という JavaScript の派生言語で書かれていま

[*1] https://pokemonshowdown.com/

[*2] https://github.com/smogon/pokemon-showdown

す。TypeScript は JavaScript に型定義機能を追加した言語で、コンパイルすることで JavaScript に変換されます。JavaScript は Web ブラウザ上で動作する言語として広く用いられていますが、ここでは node.js というサーバ上で動作する処理系を用います。ネットワーク通信を行わず、開発用のパソコン単体でオフラインでシミュレータを動作させることができます。本当は AI 側でも TypeScript の型定義を利用したかったのですが、特殊なコンパイルプロセスが用いられていたため、これを呼び出す AI 側は型定義を利用せず通常の JavaScript で記述することとしました。

人間ではなく AI からシミュレータを操作するにあたり、主に 2 種類の手段が考えられます。1 つはシミュレータの内部状態を取り出して利用する方法で、最大限の自由度がある一方、データ構造について深く理解する必要があります。もう 1 つは GUI との通信に用いられる通信プロトコルをそのまま用いる方法で、あらゆる情報を得られるわけではないものの、ソースコードに最低限の説明ドキュメントが添付されています。今回は後者を選択しました。

2.1.3 先行研究での利用

Pokémon-Showdown を AI によるポケモンバトルに利用する先行研究として、IEEE's 2017 Conference on Computational Intelligence in Games で開催された Showdown AI Competition [*3]が存在します。これはポケモンバトル AI を開発するコンペティション開催のために用意されたプラットフォームのようです。この研究においても、通信プロトコルを解釈し、行動を選択する AI と連携させる手段が用いられています。開発当時の最新ルールである第 6 世代（X・Y）に対応するため大量のメッセージに対応する必要があったようで、通信プロトコルの解釈だけで千行近いソースコードが記述されています [*4]。コンペティション開催のお知らせは Web 上で閲覧できるものの、参加者や結果に関する情報がまったく見つからないため、残念ながら実際に開催されたのかどうかを含め確認することはできませんでした。なお本書との大きな違いとして、パーティはランダムに生成されたものが AI に渡されるというルールになっています。パーティ生成の技術開発は不要な一方で、どんなパーティを受け取っても運用する技術が必要となります。

[*3] http://game.engineering.nyu.edu/showdown-ai-competition/

[*4] https://github.com/scotchkorean27/showdownaiclient/blob/19f92bb0ccc4cfdea72456934f6ec58eb127d895/interfaceLayer.js

2.1.4 プロトコルの概要

まずは先述のシミュレータの通信プロトコルについて概説します。その後、AI で必要となる各機構との接続方法について独自に開発した内容を説明します。

プロトコルはテキストベースで、チャンクという単位で転送されます。1 つのチャンクは、1 ターンで起きた出来事、プレイヤーの 1 回の行動の要求、プレイヤーが選択した行動 1 つなどに対応します。

バトルを行う上で必要なチャンクを掲載します。チャンクの名称は筆者がつけたものです。#以降は筆者がつけたコメントです。AI → Sim は AI 側からシミュレータへ送るチャンク、Sim → AI はシミュレータから AI に送られるチャンクです。

▼リスト 2.1　AI → Sim start チャンク。バトル開始設定を行う。

```
>start {"formatid": "gen2customgame"} # 第 2 世代ルールを用いる
>player p1 {"name": "p1", "team": "slowpoke||berryjuice|noability|return,... # pack
したパーティ情報（後述）
>player p2 {"name": "p2", "team": "venonat||przcureberry|noability|psywave,...
```

▼リスト 2.2　Sim → AI sideupdate チャンク。プレイヤーに行動を要求する。

```
sideupdate
p1 # 対象プレイヤー。p1 または p2。
|request|{"active":[{"moves":[{"move":"Return 102","id":"return","pp":32,... # 味
方の状態を JSON で格納。
```

▼リスト 2.3　request のパラメータ。行動要求における味方の状態を表す内容。

```
{
  "active": [
    {
      "moves": [
        {
          "move": "Hydro Pump",
          "id": "hydropump",
          "pp": 8,
          "maxpp": 8,
          "target": "normal",
          "disabled": false
        },
        {
          "move": "Body Slam",
          "id": "bodyslam",
          "pp": 21,
          "maxpp": 24,
```

```
          "target": "normal",
          "disabled": false
        },
        ... # 技 4 つ分
      ]
    }
  ],
  # 瀕死に伴う強制交代では、active がなく forceSwitch: true が含まれる
  "side": {
    "name": "p1",
    "id": "p1",
    "pokemon": [ # パーティ全体の状態。
      {
        "ident": "p1: Wartortle",
        "details": "Wartortle, L55, M",
        "condition": "100/181 tox", # HP と状態異常。
        "active": true, # 場に出ているポケモンに true がつく。場に出ているポケモンが常に
リストの先頭に来ることに注意。
        "stats": {
          "atk": 125,
          "def": 144,
          "spa": 127,
          "spd": 144,
          "spe": 119
        },
        "moves": [
          "hydropump",
          "bodyslam",
          "rollout",
          "seismictoss"
        ],
        "baseAbility": "noability",
        "item": "quickclaw",
        "pokeball": "pokeball"
      },
      {
        "ident": "p1: Kabuto",
        "details": "Kabuto, L50, M",
        "condition": "136/136",
        "active": false,
        ...
      },
      {
        "ident": "p1: Slowpoke",
        "details": "Slowpoke, L50, M",
        "condition": "196/196",
        "active": false,
        ...
      }
    ]
  }
}
```

▼リスト 2.4　AI → Sim choice チャンク。プレイヤーの行動を与える。

```
>p1 move 2 # 2 番目の技を選択したことを示す。交代の場合は switch 1 のようになる。
```

▼リスト 2.5　Sim → AI update チャンク（バトル開始時）。

```
update
|player|p1|p1||
|player|p2|p2||
|teamsize|p1|3
|teamsize|p2|3
|gametype|singles
|gen|2
|tier|[Gen 2] Custom Game
|rule|HP Percentage Mod: HP is shown in percentages
|
|start
|split|p1 # 次の行はプレイヤー 1 にのみ見えることを示す（たとえば片方のプレイヤー向けのメッ
セージでは HP が実数で出て、全体向けにはパーセンテージのみが出るという設定が存在する）
|switch|p1a: Slowpoke|Slowpoke, L50, M|196/196 # 最初のポケモンが繰り出された
|switch|p1a: Slowpoke|Slowpoke, L50, M|196/196
|split|p2
|switch|p2a: Venonat|Venonat, L50, M|166/166
|switch|p2a: Venonat|Venonat, L50, M|166/166
|turn|1 # 次のターン番号
```

▼リスト 2.6　Sim → AI update チャンク（1 ターンの経過を示す）。

```
update
|
|split|p2
|switch|p2a: Ursaring|Ursaring, L50, M|196/196 # ポケモンの交代
|switch|p2a: Ursaring|Ursaring, L50, M|196/196
|move|p1a: Wartortle|Body Slam|p2a: Ursaring # 技の発動
|split|p2
|-damage|p2a: Ursaring|159/196 # ダメージを受けた後の状態
|-damage|p2a: Ursaring|159/196
|-status|p2a: Ursaring|par # 状態異常の発生
|split|p1
|-damage|p1a: Wartortle|170/181 tox|[from] psn # どくによるダメージ
|-damage|p1a: Wartortle|170/181 tox|[from] psn
|
|upkeep
|turn|3
```

▼リスト 2.7　Sim → AI end チャンク。バトルが終了したことを示し、勝者などの情報が付随する。

```
end
{"winner":"p1","seed":[20308,19532,32518,7845],"turns":14,"p1":"p1","p2":"p2",...
```

バトルは次のような手順で進行します。

1. AI → Sim start チャンク。バトル開始設定を行う。
2. Sim → AI sideupdate チャンク。プレイヤーに最初のターンの行動を要求する。
3. Sim → AI update チャンク（バトル開始時）。
4. AI → Sim choice チャンク。プレイヤーの行動を与える。
5. ポケモンが瀕死になった場合: Sim → AI sideupdate チャンク。プレイヤーに強制交代で繰り出すポケモンを要求する。
6. Sim → AI sideupdate チャンク。プレイヤーに次のターンの行動を要求する。
7. Sim → AI update チャンク（1 ターンの経過を示す）。バトル終了時以外は 4. へ移動。
8. バトル終了時: Sim → AI end チャンク。

少し変わっているのですが、あるターンの経過が update チャンクで送られるより前に、なぜか sideupdate でその次のターンの行動を要求してきます。sideupdate を受信した瞬間には、まだ直前の出来事がわからないため行動の判断ができません。そのため update チャンクを受け取った後に思考を行い、choice チャンクを送る必要があります。

2.1.5 AI との接続

シミュレータと AI との接続の実装について紹介します。全体像を図 2.1 に示します。

シミュレータは JavaScript で記述されていますが、AI 側は強化学習フレームワーク等が充実している Python で記述します。これらは同一プロセスで動作させることができないので、プロセス間通信を用いました。リスト 2.8 に示す、シミュレータの入出力となる文字列を標準入出力に転送する簡単な JavaScript プログラムを実装し、これを Python 側から起動します。

▼リスト 2.8 Python 側から起動される JavaScript コード。シミュレータを起動し標準入出力と接続する。

```
// 標準入出力によるプロセス間通信によりシミュレータを公開
// chunk 単位を json シリアライズして 1 行で送受信

const bs = require('../Pokemon-Showdown/.sim-dist/battle-stream');
const BattleStream = bs.BattleStream;

// BattleStream というクラスがシミュレータを内包する Showdown のクラス
const stream = new BattleStream({ debug: false, keepAlive: true });
```

```
const reader = require('readline').createInterface({
    input: process.stdin,
    output: process.stdout
});
// プロセスの stdin=Python 側の出力から受け取った文字列を BattleStream に書き込み
reader.on('line', function (line) {
    stream.write(JSON.parse(line));
});

// BattleStream から受け取った文字列をプロセスの stdout=Python 側に書き込み
(async () => {
    let chunk;
    while (chunk = await stream.read()) {
        process.stdout.write(JSON.stringify(chunk) + '\n');
    }
})();
```

次に Python で記述する AI 側です。

まず、シミュレータと直接通信するのは Sim クラスです。Sim クラスでは、リスト 2.8 のコードを実行する node.js プロセスを開始し、標準入出力を通じて通信します。最初に対戦するパーティをシミュレータに送信し、その後はシミュレータからチャンクを受け取るごとに動作します。チャンクを受信すると、各プレイヤーごとに内容を振り分け、BattleStreamProcessor クラスの各プレイヤーごとのインスタンスへとその内容を送ります。

BattleStreamProcessor クラスは、各プレイヤーを代表します。Sim クラスから、リスト 2.2 やリスト 2.6 のようなチャンクを受け取ります。プレイヤーから見える範囲のバトルの状況を BattleStatus クラスのオブジェクトとして保持し、チャンクの内容に応じて更新します。ごく一部を抜粋してリスト 2.9 に示します。この例では、天候が変化したことを示すチャンク内のメッセージに反応して、BattleStatus オブジェクトの天候の属性を書き換えています。実際には、ダメージ発生・ポケモン交代・ランク変化など 20 種類程度のメッセージに対応しています。

▼リスト 2.9 バトル中の出来事に関するメッセージを受け取り、保持しているバトル状態を更新するコード（抜粋）。

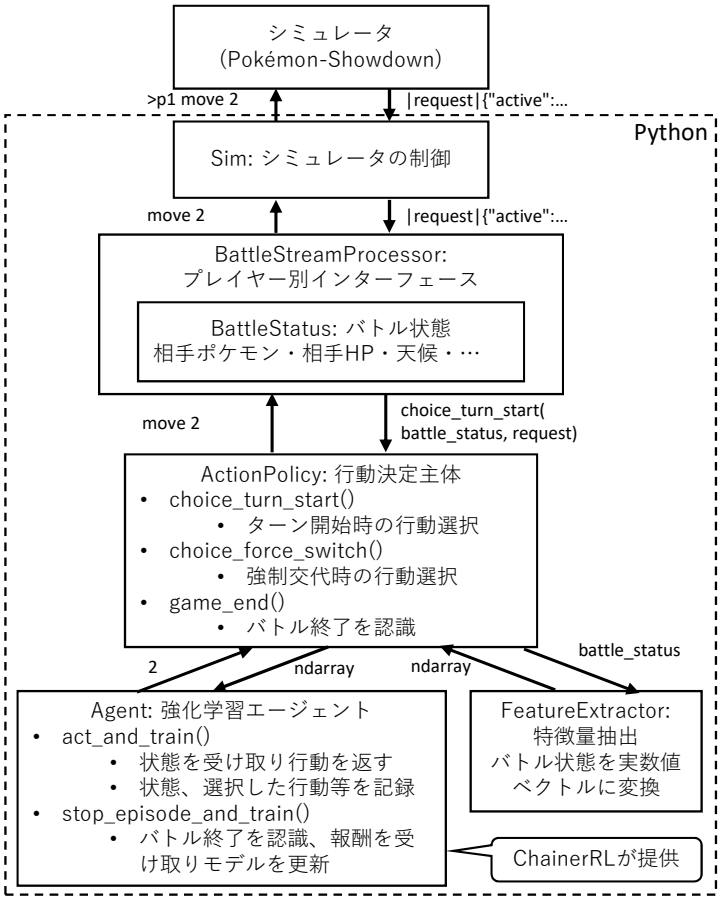
```
class BattleStreamProcessor:
    def _handle_weather(self, msgargs: List[str]) -> Optional[str]:
        # 天候の開始/終了（天候のところに none）
        # |move|p1a: Xatu|Sunny Day|p1a: Xatu
        # |-weather|SunnyDay
        # SunnyDay,RainDance,Sandstorm,none
        self.battle_status.weather = msgargs[0]
        return None

class BattleStatus:
```

```
    side_friend: str  # 自分側の side ('p1' or 'p2')
    side_party: Party  # 自分側のパーティ
    weather: str  # 天候（なしの時は WEATHER_NONE='none'）
```

ActionPolicy クラスは、状態を受け取って行動を決定する主体です。BattleStatus クラスから、図に示すような 3 つのメソッドが呼び出されます。アルゴリズムに応じてサブクラスが実装されており、ランダムに行動する RandomPolicy や、強化学習による方策モデルに基づき行動する RLPolicy があります。

RLPolicy クラスでは、強化学習ライブラリには ChainerRL[*5]を用いており、Agent(chainerrl.agent.Agent) のサブクラスである強化学習エージェントを用いて行動を選択します。act_and_train は現在のバトルの状態を受け取り、現在の方策モデルに基づく行動を返すとともに、将来モデルを更新するために状態・選択した行動等を内部状態として記録します。バトル終了時に stop_episode_and_train が報酬（バトルの勝敗）を受け取り、バトルの履歴を用いてモデルが更新されます。なお、学習が完了したモデルの評価の際には act_and_train の代わりに act、stop_episode_and_train の代わりに stop_episode が呼ばれるようになっています。Agent は状態を表す実数値ベクトル（特徴量）を受け取るため、BattleStatus オブジェクトを FeatureExtractor によって変換します。強化学習に関する設計は後述します。このような流れで、バトルの状態を AI に与え、AI の判断をシミュレータに返すことでバトルが進行します。

[*5] https://github.com/chainer/chainerrl



▲図 2.1　AI による行動選択システムの呼び出し構造

永遠に終わらないバトル

実験中、シミュレータがメモリを数 GB 消費し、最終的にはメモリ不足でクラッシュするという事例がありました。どうやらバトルの条件によっては極めて長いターンかかり、履歴情報がたまり続けてしまうようです。長くなりそうなポケモンバトルの状況について考えてみました。ツボツボというポケモンは、攻撃力が非常に低い一方、防御力が非常に高いという特徴があります。ツボツボ（レベル

50）同士が最後の 1 匹として対面し、技の PP も切れてしまってわるあがきを打ち続ける場合を考えます。ダメージ計算ツールによれば、ツボツボの HP は 126 で、わるあがきの通常のダメージは最高 3、急所に当たった場合でも最高 5 となりました。反動で与えたダメージの 1/4 を攻撃側もダメージとして受けますが、それを合わせても 1 ターン当たり最大 6 しかダメージを受けません。ツボツボの持ち物がたべのこしだと、毎ターン HP を 7 回復します。この状況に陥ると、永遠にバトルが終わらないということになります。この問題を回避するため、バトルが 100 ターンに達したら引き分けとするように実装しました。シミュレータに`>forcetie` というチャンクを送ることで、引き分けとしてバトルを終了させることが可能です。

このような事態を避けるためか、第 4 世代（ダイヤモンド・パール）以降ではわるあがきを使うと、与えたダメージではなく使用者の最大 HP の 1/4 を反動として受ける仕様に変更されています。

2.2 パーティの生成

シミュレータはポケモンバトルを司る機構であり、パーティの生成に関しては基本的に関知しません。

シミュレータに付随した機能として、人間同士のバトルにおいて、即席でゲームバランスのとれたパーティを生成する機能が備わっています。この機能は、ポケモンの強さランクや各ポケモンに合った強力な技のテンプレートに基づいて動作します。しかしこれはどんなポケモンや技が有効かということを人間が設定したものであり、本書の目的に合致しません。本書では、ルール上存在しうる全てのパーティを生成可能とした上で、AI による評価に基づいて強力なパーティを生成します。ここでいうルールとは、イントロダクションで述べたようにパーティ内のポケモン数、レベルなどであり、そこから副次的に生じるポケモンの覚えられる技の制約です。ルールを守ったパーティ生成のためには、少なくともポケモンが覚えられる技の一覧が必要です。これはシミュレータから抽出することが可能です。

▼リスト 2.10 Pokemon-Showdown/data/mods/gen2/learnsets.js より、ポケモンが覚える技の情報。

```
let BattleLearnsets = {
        bulbasaur: {learnset: {
                bide: ["1M"],
                growth: ["1L34", "2L32"],
        ...
```

ポケモンが覚える技とその条件が列挙されています。この例では、フシギダネ（bulbasaur）ががまん（bide）を第 1 世代の技マシン（1M）で覚え、せいちょう（growth）を第 1 世代ではレベル 34（1L34）で、第 2 世代ではレベル 32（2L32）で覚えるというようなことが書かれています。このデータファイルから、ポケモンの種族ごとに覚えられる技の一覧を生成することができます[*6]。

シミュレータでは、次のようなフォーマットでパーティを扱います。

▼リスト 2.11　シミュレータ上のパーティ形式（JSON）。

```
[
  {
    "name": "marill",
    "species": "marill", # ポケモンの種類
    "moves": [ # 技
      "dynamicpunch",
      "doubleteam",
      "endure",
      "rest"
    ],
    "ability": "No Ability", # 特性（使用しない）
    "evs": { # 努力値
      "hp": 255,
      "atk": 255,
      "def": 255,
      "spa": 255,
      "spd": 255,
      "spe": 255
    },
    "ivs": { # 個体値
      "hp": 30,
      "atk": 30,
      "def": 30,
      "spa": 30,
      "spd": 30,
      "spe": 30
    },
    "item": "energypowder", # 持ち物
    "level": 5, # レベル
    "shiny": false, # 色違い
    "gender": "M", # 性別（M/F/N）
    "nature": "" # 性格（使用しない）
  },
```

[*6] 進化前でのみ覚えられる技は進化前のポケモンの情報にしか掲載されていないため、情報を統合する機構も実装します。

```
  {
    "name": "sunflora",
    "species": "sunflora",
    "moves": [
      "toxic",
      "cut",
      "razorleaf",
      "rest"
    ],
    ... # 2 匹目以降の情報
]
```

パーティの生成する場合は、上記フォーマットにポケモンの種類、技、持ち物、性別を埋めることで実装できます。シミュレータのバトル開始時に与える場合、「pack」した形式で与える必要がありますが、これはシミュレータの Dex クラスの packTeam メソッドで簡単に行うことができます。

技の両立

あるポケモンが覚えられる技のリストからランダムに 4 つ技をピックアップした場合、実際にはその組み合わせを覚えさせられないという場合があります。細かいことではあるのですが、その状況の 1 つとしてタマゴ技と第 1 世代限定の技マシンの組み合わせがあります。タマゴ技は、第 2 世代でタマゴから生まれたポケモンだけが覚えられる技のことです。タマゴ技を覚えたポケモンは第 2 世代で生まれる必要があります。そのポケモンをタイムカプセル（特殊な通信交換で、第 1 世代と第 2 世代でポケモン交換を行うモード）で第 1 世代に送ることで第 1 世代にしかない技マシンで技を覚えさせることができるのですが、ここでタマゴ技が第 2 世代限定のものだと、第 1 世代にその技が実装されていないためタイムカプセルで送ることができません。実例として、フシギダネのタマゴ技「しんぴのまもり」（第 1 世代に登場しない技）と第 1 世代にしかない技マシンで覚える技「つるぎのまい」は両立しません。別の条件として、タマゴ技の遺伝経路の問題により、カラカラに「つるぎのまい」と「ほろびのうた」を両立させることはできません。このような状況が発生しうるため、技 4 つを決めた後、技の組み合わせをチェックする必要があります。このチェックは、シミュレータに TeamValidator という機能があり、これを呼び出すことで実行することができます。この機能は結構細かくて、個体値と性別のミスマッチなども確認してくれます。しかしメスしかいないポケモンをオスとしてもエラーにならないなど、少し抜けた挙動もあ

るようです。

2.2.1 パーティ評価関数によるパーティの最適化

ルール上存在できるパーティの生成方法がわかったので、次は強力なポケモンや技を備えたパーティを自動的に生成するアルゴリズムについて説明します。強力なパーティを現実的な計算コストで生成する手法として、第 2 巻で「パーティ評価関数」を用いる手法が有効でした。本巻では、金銀ルールにおいて追加されたパーティの要素である、持ち物を追加し手法を拡張します。

まず、パーティ評価関数について再掲します。

パーティ評価関数は、パーティ X を入力とし、その強さを推定する関数 $R(X)$ です。このような関数が存在すれば、$R(X)$ が大きな値をとるような X を探索することで、強力なパーティを生成することができます。

強さを推定する関数 R について検討します。バトルで強いパーティを生成したいので、直接的にバトルで測定するのが妥当な方法だと考えられます。バトルにおける強さの指標としてもっとも単純なのは勝率ですが、強さに大きな開きがあるときに扱いづらいという問題があります。ランダムに生成したパーティと最適化されたパーティでは圧倒的な戦力差があり、ランダムなパーティに対する勝率だと 99% というような値になってしまい最適化されたパーティ間の強さ比較が難しくなります。一方で、パーティの強さごとに対戦相手を変えてしまうと勝率という指標での比較ができなくなってしまいます。この問題への対処として、チェスなどの競技のプレイヤーの強さを表す指標として使われているイロレーティング（Elo rating）を用います。

イロレーティングでは、プレイヤーそれぞれの強さが実数値（以下ではレートと呼ぶ）であらわされます。そして、プレイヤー間のレートの差が「そのプレイヤー間での」期待される勝率を表します。プレイヤーの平均レートは 1500 で、プレイヤー A のレートがプレイヤー B より 200 高いとき、プレイヤー A がプレイヤー B に 76% の勝率であると期待されます。数式で書くと、プレイヤー A のレートが R_A、プレイヤー B のレートが R_B で、プレイヤー A の B に対する勝率が W_{AB} であるとき、これらは $W_{AB} = 1/(10^{(R_B - R_A)/400} + 1)$ という関係があるとみなします。実験ではレート 2100 を超えるようなプレイヤー（パーティ）が出てきます。レート 1500 のパーティとの対戦勝率だと 97% というような値になり、レート 2150 でも 2050 でもほとんど差が出ないような状況になります。しかし近いレート同士のパーティで対戦させ、上記の計算式が満たされるようにレートを算出すれば、大きな戦力差があるパーティでも一直線の指標で並べ

ることができます。本書では、ベンチマーク用にランダム生成したパーティを相互対戦させ、レートを算出します。パーティの強さにばらつきがあるため、レートが 900〜2000 程度で散らばります。そしてベンチマーク用パーティのレートを固定した状態で、測定したいパーティを追加して対戦させ、整合性が取れるようにレートを計算することで同一基準での強さ指標を計算しています。このようにしてバトルによって強さを測定する関数を $R_b(X)$ とします。

レートの計算をするためには様々な相手と戦って勝敗の統計を取る必要があり、バトル 1 回だけで計算はできません。バトル 1 回に数 ms 程度かかるため、$R_b(X)$ の 1 回の評価には数秒程度を要することになります。$R_b(X)$ が大きくなるような X を計算するには、様々な X に対して $R_b(X)$ を計算する必要があり、試行回数を稼ぎにくいと良い結果を得ることができません。バトルで強いパーティを生成することが目的なので、バトルで強さを測定するのがもっとも正確なのですが、この問題に対処するため、もっと高速な手法でこの結果 $R_b(X)$ を近似することを考えます。そこで、パーティから特徴量を抽出し、特徴量から強さを算出する回帰モデル $R_r(X)$ を学習するという機械学習ベースの手法を取り入れることにしました。

パーティにおける特徴量とは、パーティの強さを表すために役立ちそうな情報を取り出したものです。例えば、パーティにフーディンが含まれているかどうか、コイキングが含まれているかどうか、ふぶきが含まれているかどうか、などが考えられます。フーディンが含まれていれば強い傾向にあるし、コイキングが含まれていれば弱い傾向にあるというような計算ができそうです。一番単純な特徴量として、パーティに特定のポケモンや技が含まれるか否か、という方法が考えられます。もう少し表現力を上げようと思えば、どくどくとかげぶんしんを両方覚えているポケモンがいるかどうか、のようなコンボとして使える技の組み合わせを評価対象にすることが考えられます。2 つの要素が同時に成立しているかどうかという判定を行うことで表現力の向上を狙っています。第 2 巻では、ポケモン、技、ポケモン 2 匹の組み合わせ、（1 匹が覚えている）技 2 つの組み合わせ、ポケモンと技の組み合わせの 5 種類を特徴量として使いました。これらをあわせて「2 技関係」特徴と呼んでいます。数値的には、パーティ内で特定の要素が出現しているか否かでベクトルの要素を 0 または 1 に設定することとしました。図 2.2 に具体的なパーティからの抽出例を示します。ポケモンは 251 種類、技は 251 種類あります。ポケモンの存在を表す P 特徴は 251 次元、技の存在を表す M 特徴は 251 次元、ポケモン 2 匹の組み合わせの存在を表す PP 特徴は $251 \times 250 \div 2 = 31375$ 次元、技 2 個の組み合わせの存在を表す MM 特徴は $251 \times 250 \div 2 = 31375$ 次元、ポケモンと技の組み合わせの存在を表す PM 特徴は $251 \times 251 = 63001$ 次元となります。単純にすべての組み合わせを網羅するため、実際には「ゲンガー・キノコのほうし」のようにポケモンと覚えられない技の組み合わせも存在しています。

入力パーティ

ポケモン	技
ギャラドス	はかいこうせん・なみのり・ れいとうビーム・ハイドロポンプ
ケンタロス	ふぶき・のしかかり・じしん・かいりき
ゲンガー	ナイトヘッド・１０まんボルト・ サイコキネシス・さいみんじゅつ

⇩ パーティ特徴量抽出

P特徴	コイキング	ギャラドス	カビゴン	ケンタロス
値	0	1	0	1

M特徴	なみのり	じばく	じしん	かたくなる
値	1	0	1	0

PP特徴	ギャラドス・ ケンタロス	ギャラドス・ サンダー	マルマイン・ サンダース	ゲンガー・ ケンタロス
値	1	0	0	1

MM特徴	なみのり・ れいとうビーム	なみのり・ ふぶき	10まんボルト・ さいみんじゅつ	でんじは・ どくどく
値	1	0	1	0

PM特徴	ギャラドス・ なみのり	ギャラドス・ かいりき	ラッキー・ とっしん	ゲンガー・ きのこのほうし
値	1	0	0	0

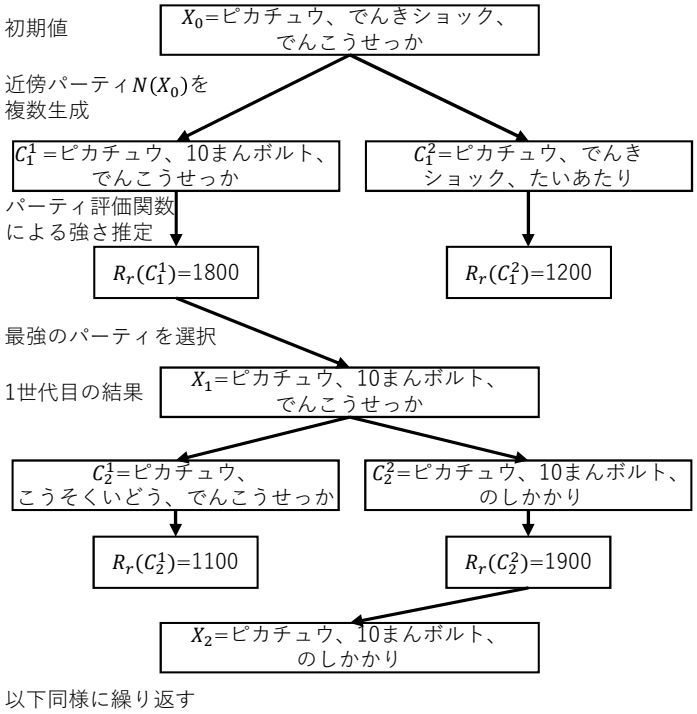
▲図 2.2　パーティ特徴量抽出の例。特徴は大きく分けて 5 種類あり、それぞれのうち 4 次元を表示。実際は合計で 126,253 次元ある。

この特徴量を入力として、強さへと回帰します。今回は単純かつ結果が解釈しやすい線形モデルを用いました。イメージとしては、フーディンは 3 点、コイキングは-2 点、ふぶきは 2 点というように、各特徴に点数をつけ、パーティに該当する点数の合計値を強さとします。この点数配分を決めるため、機械学習の一種である教師あり学習を用います。ランダムに生成したパーティを戦わせることで各パーティのレートを計算し、これを学習データとしてモデルの係数（点数配分）を学習します。学習手法は一般的な線形 Support Vector Regression を用いました。回帰の学習においては、イロレーティングを平均 0、標準偏差 1 にスケーリングした値に回帰させ、後処理で本来のスケールに変換しました。

バトル不要で強さを予測するという目標なのに、学習データとしてパーティをバトルさせてレート計算させるというのは奇妙に聞こえるかもしれません。しかし、パーティ 10,000 個のレートをバトルで計算してモデルを学習してしまえば、そのモデルを用いてあらゆる未知のパーティの強さを予測できます。もちろん誤差は出るのですが、あらゆる

ポケモン・技・持ち物の組み合わせを考えると潜在的には 10^{30} 個のパーティが存在するにもかかわらず、そのほんのわずかだけを学習対象にしてパラメータを学習してしまえば、パーティのレートを予測するのは一瞬（1ms 以下）でできるようになるため、十分に効率的だといえます。これを汎化能力といい、これを実現するのが機械学習の大きな目標です。

このようにしてパラメータが学習されたレート回帰モデル $R_r(X)$ の値が最大となるようなパーティ X を生成するために、単純な離散最適化手法である山登り法を用います。山登り法の概要を図 2.3 に示します。最初にランダムなパーティ X_0 を生成し、これを少しだけ変更した近傍パーティ候補 $N(X_0) = \{C_1^1, C_1^2, ...\}$ を生成します。そして、これを R_r で評価し、最も強いパーティを選択します。この処理を 1 世代とします。選択されたパーティを X_1 とし、次の世代では再度これを少しだけ変更したパーティ候補 $N(X_1) = \{C_2^1, C_2^2, ...\}$ を生成します。そして、これを R_r で評価し、最も強いパーティを選択します。これを繰り返すことで徐々に強いパーティへと変化させていきます。ここで、パーティを少しだけ変更する手段は 3 通り用意しています。(1) ポケモンを 1 匹完全に別のものにし、技・持ち物もランダムに設定する、(2)1 匹のポケモンを選び、その技を 1 つ選び、別のものにする、(3)1 匹のポケモンを選び、その持ち物を別のものにする。(1)・(3) が選ばれる確率はそれぞれ 10%、それ以外では (2) が選ばれ、パーティが変更されます。



▲図 2.3　山登り法の概要。

2.2.2　持ち物を考慮した特徴量の拡張

金銀ルールで追加された持ち物は、パーティ生成において技と類似した性質をもつものと考えられます。持ち物に関する特徴は文字 I で表すこととします。単純にパーティ内に特定の持ち物が存在するかどうかというのがもっとも単純な特徴でしょう。これを I 特徴と呼びます。持ち物の候補は 46 種類あります（第 3 世代以降と大きく異なるため、付録に一覧を載せてあります)。特定のポケモンが持っている時のみ有効な持ち物（カモネギに対するながねぎや、ガラガラに対するふといホネなど）があるため、ポケモン・持ち物の組み合わせである PI 特徴も有用と考えられます。また、技「ねむる」を用いて体力を回復した直後に目覚めるための持ち物「はっかのみ」(第 3 世代以降における「ねむカゴ」)のように技と持ち物の組み合わせとなる MI 特徴も考慮するとよいでしょう。

これらのことから、従来の P,M,PP,MM,PM 特徴のほかに、I,PI,MI 特徴を追加し、パーティの強さを回帰し、またそれに基づいてパーティを生成することとしました。合計特徴量次元数は 149,894 とかなり大きくなっています。

2.3 Actor-Critic 型強化学習による行動方策の学習

パーティが与えられた時、そのバトル中の行動を選択して勝利に導くのがバトル中の行動方策です。行動方策は、各ターンの初めに自分や相手の状態を受け取り、技または交代の行動を出力する関数です。

バトルは複数ターンにわたって現在の状況に適した行動をとる必要があり、最終結果として勝敗を得ます。各ターンでの正しい行動は教えてもらえないが、複数ターンの行動の結果として勝敗を教えてもらうことができ、勝率を高めることを目標とした行動方策を学習するという要求となります。このような要求にこたえるのが強化学習という分野です。強化学習は、エージェントと呼ばれる行動主体があり、環境（ポケモンバトルのシミュレータ）からバトルの状態および報酬（勝敗）を受け取り、その状態において適切な行動を環境に返すというループで成り立ちます。本巻では詳細には立ち入りませんので、雰囲気だけお伝えします。

第 2 巻までは、Q 学習ベースの強化学習、具体的には Deep Q-Network を用いていました。この枠組みでは、あるバトルの状態 s においてある行動 a をとった際の報酬の期待値を表す関数 $Q(s,a)$ を学習します。そして、与えられた状態に対して Q が最大となる行動を選択します。ここで問題となるのは、行動選択が決定的になってしまうということです。決定的というのは同じ入力に対して常に同じ出力を返すということで、じゃんけんで言えば常にグーを出すような戦略です。対戦ゲームにおいては相手に出し抜かれてしまうため、あまり適切ではない枠組みであると言えます。そこで、行動を確率的にモデリングする方策ベースの強化学習手法を用いることを考えます。この枠組みでは、状態に対する各行動の確率 $P(a|s)$ を出力するような確率的な方策モデルを学習します。このようなモデルがあれば、60% の確率で技を使い、40% の確率で交代をするというような方策がモデリング可能となります。そのようなモデルを学習する強化学習手法の分類として、Actor-Critic 型の手法があります。これは、$P(a|s)$ に対応する方策モデル（行動を決める Actor）とともに、その方策の良し悪しを評価するモデル（批評を行う Critic）も学習を行い、学習をうまく進める仕掛けになっています。この分類に含まれる具体的なアルゴリズムには、A3C[*7]、ACER[*8]などがあります。強化学習が上手く進むための重要

[*7] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. Proceedings of The 33rd International Conference on Machine Learning, PMLR 48:1928-1937, 2016.

[*8] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu and Nando de Freitas. Sample Efficient Actor-Critic with Experience Replay. 5th International Conference on Learning Representations, 2017.

な条件として、学習サンプル間の相関ができるだけ低い方が良いということが知られています。学習サンプルというのは各ターンにおける状態と、その時とった行動および最終的な勝敗によって構成されます。バトルでは複数ターンにわたり同じ相手と対面することが多く、学習サンプル間に相関が高い状況となります。これを回避する手段として、A3C では同時に複数のプロセスで別々のエピソード（バトル）を行うことにより相関の低いサンプルを取り入れます。一方 ACER では過去のバトルの履歴をたくさん覚えておいて、そこからランダムに抽出した状態を用いてモデルを更新するという手段を用いています。Deep Q-Network が成功した要因の一つである Experience Replay という仕組みを Actor-Critic 型の学習手法でも利用できるようにしたのが ACER の特徴です。本書では、家庭用の 4 コア程度の PC では A3C の期待する条件が満たされないと考え、シングルスレッドでの学習に適した ACER を用いました。

ただ、将来的な方向性としては確率的な行動をすべきではあるのですが、現在の実験結果の範囲では相手を出し抜くと言ったほどの戦略が出てきていないため、効果の程ははっきりしません。今後、手法の持つ特徴と学習結果の関係を論じられれば面白いでしょう。ただ、AI 分野の著名な組織が頻繁に強化学習手法を提案しており、Atari（大昔のアメリカのテレビゲーム）において定量評価しているものの、具体的なプレイ内容について触れられていることはほとんどないことを考えると、エージェントの行動に分かりやすい違いを見出すのは難しいのかもしれません。

ポケモンバトルにおけるエージェントの入出力について説明します。

まず出力ですが、ポケモンを 3 匹ずつ所持して戦う場合、現在場に出ているポケモンの技 4 種類のほか、控えのポケモンに交代する行動があります。自分以外の 2 匹（瀕死になっていなければ）それぞれを選択できるため交代は 2 種類ということになります。ここで、行動 6 種類を 0~5 の整数に対応させたとすると、出しているポケモンが何かによってその整数と行動の意味が変わってしまうため、学習が難しくなることが懸念されます。どのポケモンが場に出ているかを認識していなくても行動ができてしまうため、分析も困難になります。そのため、3 匹のポケモンの技にはすべて別の行動番号を割り振ることにしました。

さらに、ターンの最中にポケモンが瀕死になった場合に、次のターンの開始前に強制交代があります。ターン開始時の行動選択とは別の状況だと考えると、これも別の行動番号を割り振ったほうが妥当に思えます。これらのことを考えて、各ポケモンの技 4 種類と、そのポケモンに交代する場合、瀕死時にそのポケモンに交代する場合の 2 種類で、3 匹の場合 18 通りの行動が存在するという設定にしました。図 2.4 にエージェントの入出力環境を示します。バトルの状態には様々な要素がありますが、これを表現するベクトル s を生成します。そしてこれをエージェントが持つ方策関数に入力します。ここで、θ は強化学習によって最適化されるパラメータです。具体的には、行列の形をしていて、入力され

る特徴量との積を計算します。方策関数の出力として、各行動 a ごとの選択確率 $P(a|s)$ が得られます。そして、この確率に従って行動を選択し、ターンを進めます。行動 a は、ポケモン 3 種類それぞれに対し、技 4 種類か交代・瀕死時の交代の 6 種類があり、その積で 18 通りの可能性があります。ただし、ゲンガーが場に出ているときにギャラドスが持っている技を選択するような無効な出力が可能になってしまっています。このような問題に対処するため、有効でない行動の確率を強制的に 0 にするようなハックを実装しました。以前用いていた Deep Q-Network では有効でない行動を選択すると有効な行動からランダムに選んで進め、マイナスの報酬を与えることで学習により今後選ばれないようにしていました。ChainerRL のコードを改造し、リスト 2.12 のような実装で実現しています。あまりきれいな実装とは言えませんが、状況によって選択可能な行動が制約されるといった環境は典型的なベンチマークにはなく、ユーザが考慮してやる必要があるようです。

ポケモン	技1	技2	技3	技4
(1)ギャラドス	はかいこうせん	なみのり	れいとうビーム	ハイドロポンプ
(2)ケンタロス	ふぶき	のしかかり	じしん	かいりき
(3)ゲンガー	ナイトヘッド	１０まんボルト	サイコキネシス	さいみんじゅつ

自分のパーティ構成

バトルの状態

入力特徴へ変換

特徴	敵タイプ水	敵HP率	自HP率	敵状態毒
sの値	1	0.7	1.0	0

エージェント
$P(a|s;\theta)$

自分のポケモンが倒された直後に要求される交代先選択

ポケモン	技1	技2	技3	技4	交代	瀕死交代
1	-	-	-	-	0.07	-
2	-	-	-	-	0.03	-
3	0.04	0.7	0.06	0.1	-	-

出力P: 各行動aの選択確率
灰色の6つの行動がこの場面で選択可能
確率Pに従い、行動aをランダムに選択

▲図 2.4　エージェントの入出力。

▼リスト 2.12　選択できない行動（非合法手）の確率を 0 にするハック実装

```
class SoftmaxDistributionLimited(CategoricalDistribution):
    def __init__(self, logits, valids, beta=1.0):
        # logits: 各行動のスコア（値域制限なし）
        # valids: 有効な行動に 1、無効な行動に 0 を代入したベクトル
        self.logits = logits
        self.valids = valids
        self.beta = beta
```

```
        self.n = logits.shape[1]

    @cached_property
    def logits_valid(self):
        # valids を掛け算することで非合法手への backprop を防止、F.log で合法手 (1) は 0
加算、非合法手 (0) は-10000 を加算
        # 十分小さい logit なら exp の計算過程で完全に 0 となり選ばれないので大丈夫、-inf
を与えると nan が発生して厄介
        return self.logits * self.valids + (1.0 - self.valids) * -10000.0

    @cached_property
    def all_prob(self):
        with chainer.force_backprop_mode():
            return F.softmax(self.beta * self.logits_valid)

    @cached_property
    def all_log_prob(self):
        with chainer.force_backprop_mode():
            return F.log_softmax(self.beta * self.logits_valid)

    @cached_property
    def entropy(self):
        with chainer.force_backprop_mode():
            xp = backend.get_array_module(self.logits)
            return - F.sum(self.all_prob * self.all_log_prob, axis=1)

class FCSoftmaxPolicyLimited(chainer.Chain, Policy):
    """
    合法手制限がある場合に対応した Softmax policy
    観測ベクトルの先頭に n_actions 次元の binary vector がついていると仮定
    """

    def __init__(self, n_input_channels, n_actions,
                 n_hidden_layers=0, n_hidden_channels=None,
                 beta=1.0, nonlinearity=F.relu,
                 last_wscale=1.0):
        self.n_input_channels = n_input_channels
        self.n_actions = n_actions
        self.n_hidden_layers = n_hidden_layers
        self.n_hidden_channels = n_hidden_channels
        self.beta = beta

        super().__init__(
            model=MLP(n_input_channels,
                      n_actions,
                      (n_hidden_channels,) * n_hidden_layers,
                      nonlinearity=nonlinearity,
                      last_wscale=last_wscale))

    def __call__(self, x):
        h = self.model(x)
        xp = backend.get_array_module(h)
        return SoftmaxDistributionLimited(
            h,
            # 有効な行動を表している x の先頭 n_actions 次元部分を切り出し
            chainer.Variable(xp.asarray(x[:, :self.n_actions])),
            beta=self.beta)
```

$P(a|s)$ を実際に計算するモデル（コード上で MLP クラスに対応する部分）は、従来は多層のニューラルネットワークとしていましたが、今回はあえて線形モデルを学習してみました。線形モデルであれば、パーティ評価関数と同様に学習したモデルの判断基準を可視化できるのではないかと考えられます。明示的に数式を記述すると、入力特徴量 x（68 次元のベクトル）に対し、$y = Wx + b$ を計算します。W（18×68 次元の行列）、b（18 次元のベクトル）が強化学習により最適化される要素 θ です。y はベクトルで、1 要素が 1 行動の優先度のようなものに対応します。大きな値が出ている次元に対応する行動を選ぶべきという判断がなされているということです。これを確率分布の形にするため softmax 関数を用います。softmax 関数は $p_i = \exp(y_i)/\sum_j \exp(y_j)$ という計算式です。y_i はベクトル y の i 番目の要素（スカラー値）を表します。softmax 関数により、$\sum_i p_i = 1$ となり、各行動を選択する確率の形になります。

▼表 2.1　入力特徴。自分/相手は、どちら側のパーティの情報を入力として与えるか。両方の場合は次元数が倍となる。

特徴	次元数	自分/相手	説明
有効な行動	18	自分	現在どの行動がとれるのかを表す 有効な行動に該当する次元に 1 を設定
生存ポケモン数	1	両方	瀕死でないポケモン数/全ポケモン数
ポケモンタイプ	17	相手	場に出ているポケモンのポケモンのタイプ （ノーマル・水・…）に該当する次元に 1 を設定
HP 残存率	1	両方	場に出ているポケモンの現在 HP/最大 HP
状態異常	6	両方	場に出ているポケモンの状態異常（どく・もうどく・ まひ・やけど・ねむり・こおりのうち 該当次元に 1 を設定)
ランク補正	6	両方	場に出ているポケモンのランク補正（こうげき・ ぼうぎょ・とくこう・とくぼう・すばやさ・命中・ 回避それぞれ、ランク/12+0.5 を設定）
天候	3	-	場の天候（はれ・あめ・すなあらし）に 該当する次元に 1 を設定

表 2.1 に入力特徴の全体図を示します。まず先頭にどの行動が有効であるのかを示すベクトルを代入しています。思考にはあまり役立ちませんが、リスト 2.12 で示したように、このベクトルを切り取ってモデルの出力の加工に用いて、無効な行動をとる確率を強制的に 0 にしています。これはすべての情報を網羅できているとはいえないことに注意してください。例えばやどりぎのたねを受けた状態というのは表現されていませんし、過去に相手が出してきた技の履歴情報もありません。めったに使われない次元があることで学習

が難しくなるため、学習コストと表現力のトレードオフへの対処としてこのような設計になっています。相手のポケモンについても、一意に特定できる図鑑番号（251 次元）ではなく、タイプ（17 次元）にしています。これは初代ルールの際の設計を引き継いだものですが、第 2 世代では同じタイプでも異なる特徴を持ったポケモンが多数いる（特にノーマルタイプのケンタロス・カビゴン・ハピナスなど）ため、議論の残るところです。自分のポケモンのタイプや持ち物については特徴量としては与えていません。強化学習において自分のパーティは固定しているため、勝敗情報から相手のタイプとの相性に応じて有効な行動を選べると想定しているためです。

報酬については、バトル終了時に勝ち＝+1、負け=-1 という値だけを与えました。ソフトウェア実装の都合で第 2 巻のような各ターンのダメージに基づいた報酬などは与えられていませんが、これらを追加すればより学習がうまく進むことが期待できます。

第 3 章

実験

3.1 パーティの生成

まず、強力なパーティを生成するために必要なパーティ評価関数の学習を行います。

パーティを 10,000 個生成し、相互に対戦させることでそれぞれにレートを付与します。対戦中の行動はランダムで、技と交代両方が選べる場合は 20% の確率で交代を選ぶようにしました。技 4 種類、交代最大 2 種類の中での選択は同じ確率です。

学習データとなる、パーティとレートの組み合わせ例を表 3.1 に示します。

特徴量について、一部のみを用いた場合と全部用いた場合を比較しました。Support Vector Regression によるレートの回帰を行います。また、学習データ（1 パーティが 1 サンプル）の量を $\{100, 1000, 10000\}$ の間で変化させ、学習データ量と精度についても比較します。ハイパーパラメータは正則化の強さを表す C のみです。ハイパーパラメータ調整のため、5-fold cross validation （学習データの 1/5 を精度評価用のデータとして分離、分け方を 5 通り行い精度の平均を計算）を行いました。実装には scikit-learn[*1]の `sklearn.svm.LinearSVR` を用いました。特徴量次元数が 149,894 と大きく、1 要素を 64bit の浮動小数点数で保持すると 1 サンプルあたり 600kB ほどになります。単純に 10,000 サンプル分をメモリに保持しようとするとそれだけで 6GB 必要で、普通のパソコンだと厳しいです。今回の特徴量はほとんどの次元の値が 0 であり、このようなデータはスパース行列という形式で保持するとコンパクトになります。スパース行列は、0 でない要素の値とその出現位置だけを保持するデータ形式です。スパース行列を扱うライブラリの一つに scipy[*2]があり、`sklearn.svm.LinearSVR` は scipy が提供する `scipy.sparse.csr_matrix` 形式のスパース行列を展開せずにそのまま計算に使うことが可能です。スパース行列

[*1] `https://scikit-learn.org/`

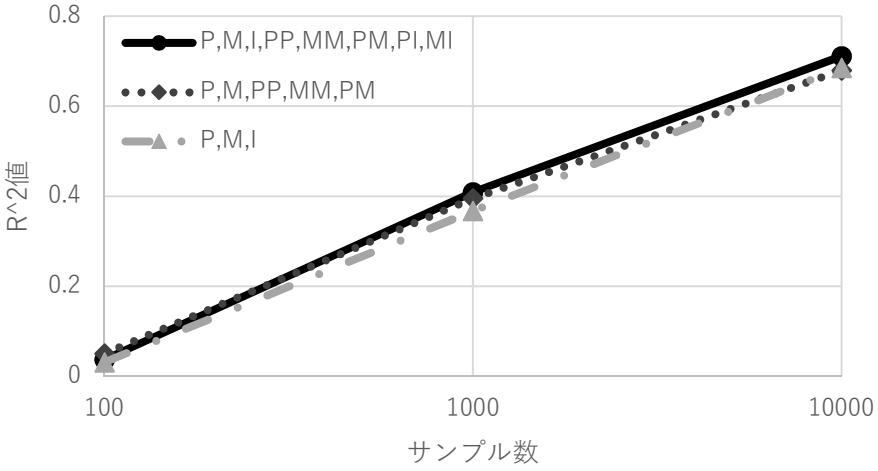
[*2] `https://www.scipy.org/`

▲表 3.1　学習データ例

ポケモン	LV	持ち物	技1	技2	技3	技4	レート
ニドキング	50	おうごんのみ	バブルこうせん	おんがえし	にらみつける	にどげり	
フォレトス	55	ひかりのこな	ギガドレイン	ずつき	すなあらし	すてみタックル	2034
オクタン	50	でんきだま	どろかけ	ちょうおんぱ	なみのり	いばる	
デリバード	50	ピントレンズ	ねむる	どくどく	プレゼント	そらをとぶ	
ケンタロス	55	おうじゃのしるし	ねむる	じしん	はかいこうせん	おんがえし	2027
スイクン	50	きせきのみ	こらえる	ねむる	おんがえし	いわくだき	
シャワーズ	55	どくバリ	あまごい	ハイドロポンプ	れいとうビーム	なみのり	
ポリゴン２	50	ながねぎ	あくむ	１０まんボルト	バリアー	かみなり	2020
ランターン	50	くろおび	いびき	はかいこうせん	ハイドロポンプ	のろい	
オーダイル	55	おうごんのみ	げんしのちから	どろかけ	なみのり	れいとうパンチ	
カイロス	50	まがったスプーン	のしかかり	はかいこうせん	どくどく	すてみタックル	2017
ニョロゾ	50	メタルパウダー	こらえる	サイコウェーブ	はらだいこ	いばる	
キングドラ	55	やわらかいすな	ふぶき	りゅうのいかり	まもる	ハイドロポンプ	
ゴローン	50	ピンクのリボン	カウンター	こらえる	じばく	おんがえし	2017
ランターン	50	メタルコート	１０まんボルト	あやしいひかり	スパーク	かみなり	
ヨルノズク	55	くろいメガネ	ふきとばし	ねごと	メロメロ	さいみんじゅつ	
ゼニガメ	50	きせきのみ	ねむる	あまごい	こうそくスピン	こらえる	876
トランセル	50	はっかのみ	たいあたり	いとをはく	かたくなる		
ヒマナッツ	55	まがったスプーン	こらえる	まもる	ねごと	はねる	
コクーン	50	でんきだま	いとをはく	かたくなる	どくばり		876
ププリン	50	りゅうのウロコ	はなびらのまい	みきり	あまごい	フラッシュ	
ププリン	50	にがいきのみ	てんしのキッス	のろい	どろかけ	みきり	
ドガース	55	ピンクのリボン	がまん	みがわり	のろい	ねごと	844
メノクラゲ	50	メタルコート	ねごと	やつあたり	メロメロ	からみつく	
リザードン	50	ぎんのこな	みがわり	メロメロ	いびき	にほんばれ	
ラッキー	55	きのみジュース	リフレクター	のろい	みがわり	しっぽをふる	823
モルフォン	50	せんせいのツメ	ちょうおんぱ	やつあたり	みやぶる	ねんりき	
ケーシィ	55	アップグレード	のろい	やつあたり	にほんばれ	バリアー	
ヒマナッツ	50	のろいのおふだ	いばる	かげぶんしん	ねごと	はねる	808
タマタマ	50	おうじゃのしるし	ねごと	だいばくはつ	どくどく	やつあたり	

を用いることで、学習プロセス全体のメモリ使用量を 200MB に抑えて処理することが可能となりました。メモリを削減する別の解としては、学習データを細かく分割して逐次的に学習アルゴリズムに投入するオンライン学習方式もあります。深層学習では基本的にオンライン学習方式しか使われないため、最近機械学習を学び始めた人には当たり前の方式かもしれません。

定量的な精度を R^2 値によって比較した結果を図 3.1 に示します。値が大きい（1 が最大、常に 0 を出力するモデルでは 0 となる）ほど回帰精度が高いことを示します。



▲図 3.1　特徴量ごとの、パーティのレート回帰精度。

大きな差ではないものの、全ての特徴量を利用する場合が最も良いことがわかります。また、データ量の増加に従って明らかに精度が向上しています。計算コストの問題で 10,000 サンプルしか用意できていませんが、さらに増加させると精度が高まることが予想できます。

次に、学習されたモデルの定性的な考察を行います。

図 3.2・図 3.3 に、特徴量の各要素に対する係数を示します。種類別にテーブルを分けています。

P特徴	係数
ハッサム	0.344
ゲンガー	0.297
エアームド	0.273
ハガネール	0.273
スイクン	0.226
ゴースト	0.226
バンギラス	0.216
ムウマ	0.212
フォレトス	0.211
カイリキー	0.203
…	
トゲピー	-0.194
マグマッグ	-0.196
ヒトカゲ	-0.199
ディグダ	-0.204
オタチ	-0.215
ププリン	-0.219
コラッタ	-0.225
レディバ	-0.228
ヒマナッツ	-0.244
ピチュー	-0.261

M特徴	係数
ころがる	0.410
なみのり	0.299
ハイドロポンプ	0.291
サイコキネシス	0.286
じしん	0.283
れいとうビーム	0.267
やどりぎのタネ	0.245
かえんほうしゃ	0.241
１０まんボルト	0.241
はかいこうせん	0.240
…	
あくむ	-0.175
あまごい	-0.176
はねる	-0.177
だいばくはつ	-0.183
ひかりのかべ	-0.190
こうそくいどう	-0.191
にほんばれ	-0.193
みやぶる	-0.204
いとをはく	-0.220
じばく	-0.238

I特徴	係数
たべのこし	0.438
きせきのみ	0.068
おうごんのみ	0.067
せんせいのツメ	0.060
ひかりのこな	0.053
きのみジュース	0.051
ふといホネ	0.035
ピンクのリボン	0.025
くろいメガネ	0.025
きせきのタネ	0.021
…	
するどいくちばし	-0.026
メール	-0.028
メタルパウダー	-0.031
たいようのいし	-0.031
のろいのおふだ	-0.031
やけたきのみ	-0.033
でんきだま	-0.035
じしゃく	-0.037
おうじゃのしるし	-0.038
はかいのいでんし	-0.262

PP特徴	係数
ツボツボ・リングマ	0.048
アズマオウ・ワタッコ	0.039
ハピナス・エーフィ	0.038
ドククラゲ・エアームド	0.036
ポニータ・エアームド	0.033
ピィ・ゲンガー	0.033
グランブル・オクタン	0.032
ヤドラン・キリンリキ	0.032
フシギバナ・ノコッチ	0.032
アーボック・モココ	0.032
…	
サイドン・イノムー	-0.032
ヒトデマン・レディバ	-0.033
ケーシィ・タマタマ	-0.034
フシギバナ・ワンリキー	-0.034
ニドラン♀・スリーパー	-0.036
ピィ・ププリン	-0.036
ニドラン♀・ニドリーナ	-0.036
コラッタ・ヤドキング	-0.037
パウワウ・レディアン	-0.040
ネイティ・ヒマナッツ	-0.040

MM特徴	係数
ねむる・かみなり	0.123
サイコキネシス・おんがえし	0.112
まるくなる・ころがる	0.111
のろい・アイアンテール	0.110
ねむる・１０まんボルト	0.105
のろい・ころがる	0.103
ねむる・すなあらし	0.097
いかり・おんがえし	0.097
まもる・どくどく	0.094
おんがえし・いやなおと	0.092
…	
かげぶんしん・やつあたり	-0.102
やつあたり・ねむる	-0.103
メロメロ・ねむる	-0.104
やつあたり・まもる	-0.105
いあいぎり・いびき	-0.110
まもる・みがわり	-0.115
あまごい・ねごと	-0.123
こうそくいどう・メロメロ	-0.128
こらえる・ねごと	-0.130
いとをはく・たいあたり	-0.163

▲図 3.2　パーティ評価関数の係数 (1/2)

PM特徴	係数
エアームド・どろかけ	0.111
ヘラクロス・メガホーン	0.103
レアコイル・１０まんボルト	0.094
フリーザー・れいとうビーム	0.092
スイクン・たきのぼり	0.090
ハッサム・はがねのつばさ	0.088
レアコイル・いばる	0.088
サンダー・１０まんボルト	0.087
レアコイル・かみなり	0.087
デンリュウ・１０まんボルト	0.087
…	
アズマオウ・みずでっぽう	-0.076
スピアー・ギガドレイン	-0.079
ヒマナッツ・いばる	-0.080
イーブイ・やつあたり	-0.082
クヌギダマ・かげぶんしん	-0.084
イトマル・ソーラービーム	-0.093
レディバ・ソーラービーム	-0.096
キャタピー・たいあたり	-0.133
キャタピー・いとをはく	-0.133
ヒマナッツ・ヘドロばくだん	-0.145

PI特徴	係数
フシギバナ・たべのこし	0.055
ポリゴン・ピントレンズ	0.049
デリバード・にがいきのみ	0.047
ドククラゲ・きあいのハチマキ	0.045
ポリゴン２・おうごんのみ	0.043
サンダー・たべのこし	0.043
トゲピー・かたいいし	0.042
バクフーン・ふといホネ	0.042
ライチュウ・たべのこし	0.041
エイパム・みずたまリボン	0.041
…	
トゲチック・ピントレンズ	-0.039
マダツボミ・するどいくちばし	-0.039
ワンリキー・くろいメガネ	-0.039
ポポッコ・メール	-0.040
ケーシィ・みずたまリボン	-0.040
ププリン・ピンクのリボン	-0.043
ポリゴン２・ひかりのこな	-0.044
ライチュウ・こおったきのみ	-0.046
オタチ・おうごんのみ	-0.050
オタチ・のろいのおふだ	-0.050

MI特徴	係数
みがわり・たべのこし	0.110
おんがえし・おうじゃのしるし	0.105
どくどく・まがったスプーン	0.097
おんがえし・にがいきのみ	0.094
こらえる・ひかりのこな	0.094
れいとうビーム・たべのこし	0.093
こらえる・メール	0.092
かげぶんしん・たべのこし	0.092
にほんばれ・じしゃく	0.089
メロメロ・アップグレード	0.088
…	
メロメロ・ラッキーパンチ	-0.083
いびき・まひなおしのみ	-0.084
ねむる・みずたまリボン	-0.087
かげぶんしん・はかいのいでんし	-0.088
こらえる・まがったスプーン	-0.090
メロメロ・たいようのいし	-0.100
いびき・はっかのみ	-0.102
たいあたり・まがったスプーン	-0.104
やつあたり・とけないこおり	-0.107
メロメロ・ぎんのこな	-0.124

▲図 3.3　パーティ評価関数の係数 (2/2)

たとえば、ハッサムの係数が 0.344 であるというのは、パーティにハッサムが含まれていればレートの推定値が 0.344 向上するということを示しています。学習前にイロレーティングの値を 200 で割っているため、本来のレートで言えば 0.344 × 200=68.8 に相当します。あくまで推定値であって、高い係数を示している要素を組み合わせたパーティが最強であるという保証があるわけではないですが、強いとされるパーティが持つべき要素がわかります。

ポケモンに対応する P 特徴を見てみると、ハッサム、ゲンガーなどが強力で、進化前のピチューやヒマナッツなどが弱いということがわかります。妥当な結果であるといえます。なお、ミュウツーやルギアなどはパーティ生成の時点で除外していることに注意してください。

技に対応する M 特徴では、ころがるが最強という一見変わった結果になっています。ころがるは、岩タイプの攻撃技で、一度発動すると複数ターンにわたり同じ技を出し続け、威力が 30・60・120・240・480 と増加していきます。5 ターンの間攻撃が続けば 1 ターンあたりの威力は 186 となり、かなり高いといえます。しかし対人戦であれば岩タイプに耐性のあるポケモンに交代されてしまい、そのままの効果が発揮されることは少ないと考えられます。ころがるがここまで高い評価を得たのは、行動がランダムという条件下に特有のものであると考えられます。逆に、じばく・だいばくはつは非常に低い係数になっています。対人戦では警戒すべき技ですが、ランダムに使うと無意味な自滅を招くでしょう。このように、技の最適な運用とかけ離れた運用がなされる前提でパーティ評価関数は学習されるというのが欠点といえます。どう使おうが無意味なはねるを含まず、使い所が良ければ強力なだいばくはつを含むパーティをどうやったら生成できるかというのが今後の課題となります。

金銀ルールで新たに追加された持ち物に対応する I 特徴を見てみます。たべのこしが圧倒的に高い係数になっています。たべのこしは毎ターン HP を最大 HP の 1/16 回復する持ち物で、いつでも役に立つ強力な持ち物であると言えます。次点はきせきのみで、全ての状態異常を回復するきのみ（第 3 世代におけるラムのみに対応）です。これも役立つ場面が広いと言えるでしょう。特定タイプの技の威力を上げる持ち物は、対応するタイプの技を持っていなければ何の意味もないので、持ち物単独の評価としては低くなりがちです。意外にも下から 2 番目となったのはおうじゃのしるしです。これは攻撃後に低確率で相手を怯ませる可能性がある道具で、特にデメリットはありません。何の効果もないたいようのいしなどより低い値になった原因はよくわかりません。最下位ははかいのいでんしで、これは第 3 世代以降に同等のものが用意されなかった特殊な持ち物です。はかいのいでんしを持ったポケモンが場に出ると、直ちに攻撃力が 2 段階上昇（2 倍）し、こんらん状態になります。極めてギャンブル要素が強いうえ、特殊技で攻めるポケモンにはデメリットしかないため弱いという評価になっていると考えられます。

2 要素の組み合わせである PP 特徴、PM 特徴などは、単独での強さの合計では表せないような組み合わせの強さを表現します。ポケモンと技の組み合わせを考えても 250 × 250 程度あり、10000 程度の学習サンプル数では網羅しきれず明確な傾向は確認しづらいです。PM 特徴では、虫タイプのヘラクロスが虫タイプのメガホーンを覚えている場合など、タイプ一致技による威力上昇のファクターが見て取れます。PI 特徴では、バクフーンがふといホネを持っている場合に加点がありますが、ふといホネはカラカラ・ガラガラが持っている場合のみ効果を発揮する持ち物でありノイズであるといえます。たまたまこの学習サンプルにおいてバクフーンの技やパーティを組んだポケモンが強力で勝率が高くなり、ふといホネの効果があったと誤認しているものと考えられます。MM 特徴では、上位にまるくなる・ころがるの組み合わせがあります。ポケモンがまるくなるを一度使うと、ころがるの威力が永続的に 2 倍になるという特殊な効果が設定されており、これが反映された可能性があります。このような組み合わせは人間でも（ゲームメーカが提供した情報に基づき編集された）攻略本に掲載されている情報を見なければ気づくことは難しいです。この例では偶然この組み合わせがランダム生成されたパーティに含まれたと考えられますが、特殊な組み合わせでのみ発現する効果をランダムな探索で見つけるのは困難と考えられます。「どんなポケモンや技が強いのかを人が教えない」という根本的な条件とできるだけ矛盾しない形で、ランダムに探索しても見つけられない戦略情報を与える方法については将来課題です。

最後に、このパーティ評価関数を最大化するようなパーティを生成しました。ランダムに生成したパーティから開始し、山登り法で 1 世代 10 パーティ、100 世代パーティ評価関数の値が高くなる方向にパーティを変化させました。

別々のランダムなパーティから 100 個生成しました。そのうちランダムな 5 個を表 3.2 に示します。パーティ評価関数が最大値となるパーティは本来 1 つだけですが、山登り法が局所的な最適解を見つける手法のため、初期値の違いにより様々なパーティが生成できています。

たべのこしはどのパーティも例外なく利用しています。ポケモンについては、進化前のポケモンも多く見られ、最適化不足が感じられます。技は、おかしなものは選ばれていないという程度でしょうか。ランダムな行動を前提としているため、補助技はねむる・どくどく程度に限られます。

3.2 行動方策の学習

前節で生成されたパーティを用いて、バトル中の行動を選択する方策を強化学習手法により学習します。

強化学習アルゴリズムは、Actor-Critic 方式の 1 つである ACER を利用します。学習

▲表 3.2　パーティ評価関数を最大にするよう最適化したパーティ

ポケモン	LV	持ち物	技1	技2	技3	技4
スイクン	50	きあいのハチマキ	おんがえし	はかいこうせん	たきのぼり	ふぶき
ドククラゲ	50	まひなおしのみ	ヘドロばくだん	ギガドレイン	ハイドロポンプ	なみのり
パウワウ	55	たべのこし	ねむる	のしかかり	バブルこうせん	れいとうビーム
ヤドン	50	きのみジュース	おんがえし	サイコキネシス	じしん	どくどく
カブト	50	たべのこし	バブルこうせん	すてみタックル	れいとうビーム	なみのり
カメール	55	せんせいのツメ	ハイドロポンプ	のしかかり	ころがる	ちきゅうなげ
パウワウ	50	たべのこし	れいとうビーム	バブルこうせん	ずつき	なみのり
ナッシー	50	にがいきのみ	ころがる	サイコキネシス	はかいこうせん	おんがえし
サイドン	55	きのみジュース	ちきゅうなげ	すてみタックル	かえんほうしゃ	じしん
マグマラシ	50	ぎんのこな	だいもんじ	ころがる	かげぶんしん	どくどく
ハクリュー	50	ふといホネ	れいとうビーム	なみのり	ハイドロポンプ	バブルこうせん
エレブー	55	たべのこし	１０まんボルト	かみなり	はかいこうせん	ねむる
ウインディ	55	にがいきのみ	のしかかり	だいもんじ	おんがえし	かえんほうしゃ
ドガース	50	どくけしのみ	どくどく	ねむる	ころがる	１０まんボルト
ネイティオ	50	たべのこし	そらをとぶ	ドリルくちばし	はかいこうせん	サイコキネシス

するモデルは、先述のようにあえて線形モデルを選択しています。学習ハイパーパラメータですが、報酬の割引率は 0.99 としました。モデル更新手法（optimizer）には Adam（alpha=0.01）を利用しました。各パーティごとに独立してエージェントを学習します。学習中の対戦相手は、エージェント学習対象のパーティ群（すなわち前節でパーティ評価関数に基づいて生成されたパーティ）100 個から毎回ランダムに選ばれます。行動はランダムです。バトル 1000 回分の学習を行いました。

学習結果のエージェントの定量的な強さを測定しました。測定基準として、ランダムに生成されたパーティ 100 個をまず相互に対戦させ、そのレートを固定します。そのうえで、強化学習エージェント 100 個および、パーティ評価関数で生成されたパーティ 100 個（強化学習中に対戦相手としたものとは別にパーティ評価関数から再生成しました）をランダムに行動させるもの 100 個を混ぜて相互に対戦させ、レートを計算しました。

表 3.3 に、最も強力だったパーティを掲載します。

▲表 3.3　強化学習の結果もっとも強かったパーティ

ポケモン	LV	持ち物	技1	技2	技3	技4	レート
ヘルガー	50	みずたまリボン	おんがえし	ヘドロばくだん	はかいこうせん	かえんほうしゃ	
ハピナス	55	たべのこし	バブルこうせん	どくどく	ころがる	サイコキネシス	2106
ジュゴン	50	おうごんのみ	なみのり	ねむる	れいとうビーム	のしかかり	
ニョロトノ	50	たべのこし	ハイドロポンプ	サイコウェーブ	サイコキネシス	バブルこうせん	
ウインディ	50	おうごんのみ	ねむる	だいもんじ	はかいこうせん	のしかかり	2094
ケンタロス	55	にがいきのみ	れいとうビーム	かえんほうしゃ	じしん	おんがえし	
ブースター	50	せんせいのツメ	かえんほうしゃ	のしかかり	どくどく	ロケットずつき	
カイリュー	55	たべのこし	１０まんボルト	ふぶき	れいとうビーム	ハイドロポンプ	2052
プテラ	50	みずたまリボン	おんがえし	だいもんじ	はかいこうせん	じしん	
バンギラス	55	りゅうのウロコ	かいりき	じしん	だいもんじ	１０まんボルト	
ラッタ	50	たべのこし	れいとうビーム	はかいこうせん	かみなり	おんがえし	2046
ニョロトノ	50	ふといホネ	のしかかり	なみのり	ハイドロポンプ	ねむる	
ゲンガー	50	たべのこし	サイコキネシス	おんがえし	１０まんボルト	ちきゅうなげ	
リザードン	50	こおったきのみ	アイアンテール	だいもんじ	いわなだれ	かえんほうしゃ	2045
カイリキー	55	たいようのいし	はかいこうせん	ねむる	かいりき	じしん	

▼表 3.4　エージェントごとのレート平均

パーティ	バトル中の行動方策	平均レート
ランダム	ランダム	1500
パーティ評価関数	ランダム	1743
パーティ評価関数	強化学習	**1884**

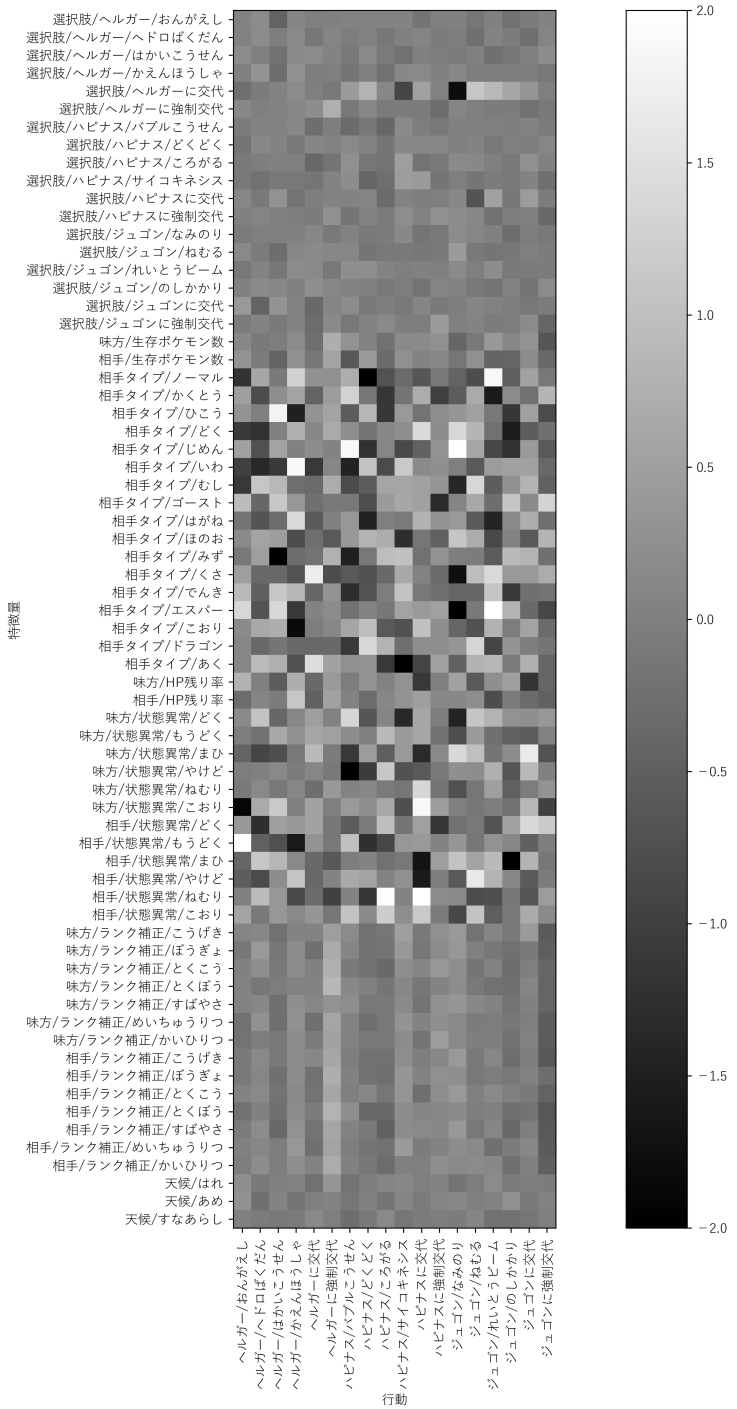
各手法で生成されたエージェント（パーティ・行動方策の組み合わせ）のレート平均を表 3.4 に示します。強化学習を用いることにより、ランダムに行動するよりも強くなっているといえます。なお、レート差 384 は、レート上位側が約 90% の確率で勝利することに相当します。同じ手法で生成されたパーティで、行動方策がランダムか強化学習されたものかでは大きな差がありません。まだまだ学習手法に改良が必要そうです。

3.3　戦略の定性的評価

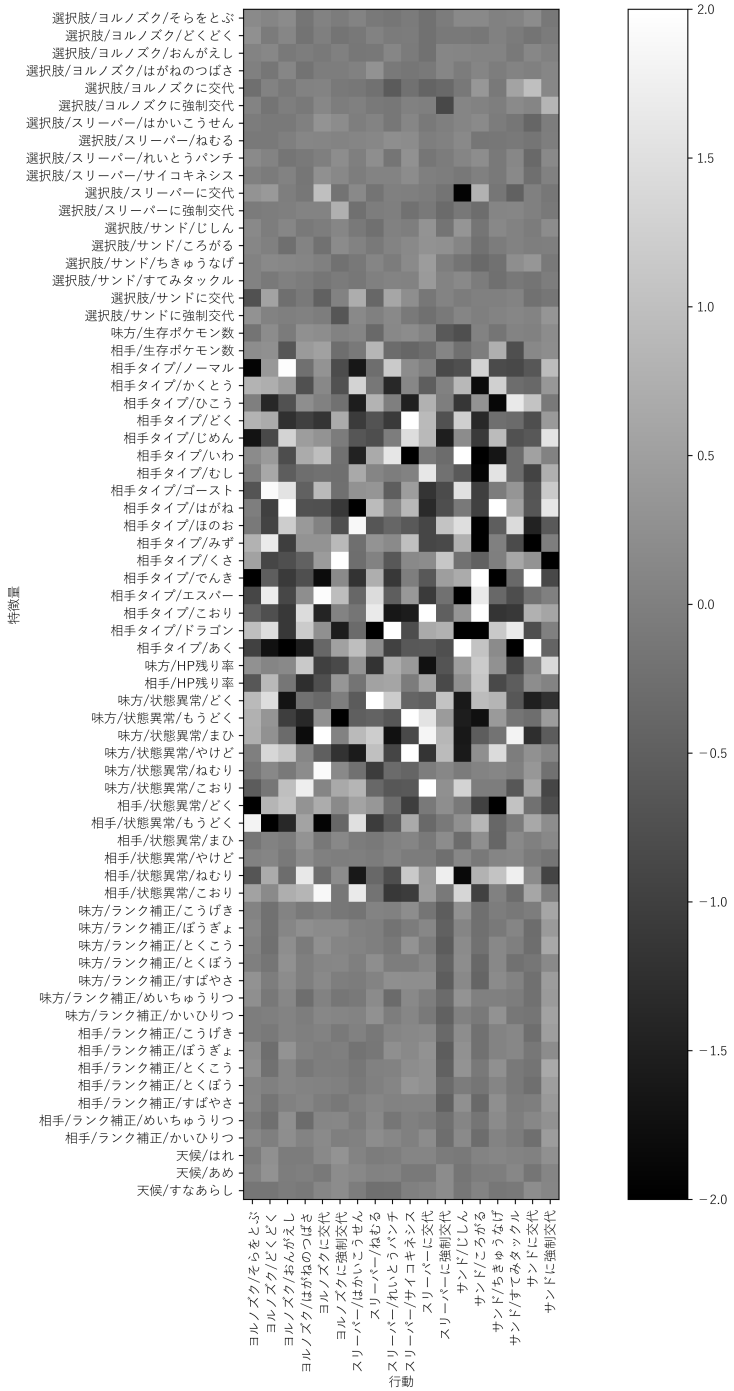
最後に、提案手法で得られた行動方策を観察します。図 3.4 は、最も強力だったパーティの行動方策モデルを可視化したものです。今回学習したモデルは、特徴量と学習で得られた行列の積が各行動の優先度として出力される線形モデルとなっており、その行列の各要素の値を色に置き換えて表示しています。たとえば、「相手タイプ/じめん」と「ハピ

ナス/バブルこうせん」が交わる点が白くなっています。これは、「相手タイプ/じめん」の特徴量の値が大きい（すなわち対面している相手のタイプが地面タイプである）時に、「ハピナス/バブルこうせん」という行動の優先度が高まる（2.0 を掛ける）ことを示しています。逆に、「相手タイプ/はがね」と「ハピナス/どくどく」の交わる点は黒くなっています。これは、相手のタイプが鋼タイプのときにどくどくを使わない（鋼タイプは毒状態にならないので）よう、その優先度が低くなる（-2.0 を掛ける）ことを示しています。このように行列の要素を読み解くことにより、強化学習でどんな思考パターンを獲得したのかを推測することが可能となります。とはいえ、じっくり見ると間違って学習されているように見える要素も多数あります。学習が失敗している場合もあれば、人間が適当に理由付けしただけで本当は違う思考パターンが正しいという場合も否定できません。

別のパーティの行動方策モデルを図 3.5 に掲載します。「相手/状態異常/もうどく」と「ヨルノズク/どくどく」の交点の値が小さく、「ヨルノズク/そらをとぶ」の値が大きくなっています。これはどのような作用を生むでしょうか？ 実際にほかのパーティと対戦したバトルログを図 3.6・図 3.7 に示します。この例では、相手が猛毒状態でなければどくどくを使い、猛毒状態になった後はそらをとぶでターン稼ぎをするという行動が見て取れます。惜しくもバトルには敗れてしまっていますが。現状、最適ではないと思われる行動も多く見られます。目視で比較的面白いと思われるバトルを探した結果としてこのようなバトルを掲示していますが、統計的にはいまいちな戦略がほとんどという印象でした。



▲図 3.4　方策モデルの係数



▲図 3.5　方策モデルの係数

パーティ1						
ポケモン	LV	持ち物	技1	技2	技3	技4
ヨルノズク	50	きのみジュース	そらをとぶ	どくどく	おんがえし	はがねのつばさ
スリーパー	50	おうごんのみ	はかいこうせん	ねむる	れいとうパンチ	サイコキネシス
サンド	55	たべのこし	じしん	ころがる	ちきゅうなげ	すてみタックル
パーティ2						
ポケモン	LV	持ち物	技1	技2	技3	技4
サニーゴ	50	ふといホネ	ころがる	どくどく	バブルこうせん	サイコキネシス
ハリーセン	50	たべのこし	ハイドロポンプ	なみのり	ヘドロばくだん	すてみタックル
ニョロトノ	55	しんぴのしずく	おんがえし	じしん	れいとうビーム	のしかかり

パーティ1		パーティ2
ターン1		
ヨルノズクを繰り出した		サニーゴを繰り出した
ヨルノズクのどくどく	→	攻撃は外れた
ヨルノズク HP206→168	←	サニーゴのころがる
ターン2		
ヨルノズクのどくどく	→	もうどくをあびた
攻撃は外れた	←	サニーゴのころがる
		毒ダメージ HP161→151
ターン3		
ヨルノズクのそらをとぶ	→	溜め
攻撃は外れた	←	サニーゴのころがる
		毒ダメージ HP151→131
ターン4		
ヨルノズクのそらをとぶ	→	サニーゴ HP131→116
ヨルノズク HP168→127	←	サニーゴのころがる
		毒ダメージ HP116→86
ターン5		
ヨルノズクのそらをとぶ	→	溜め
攻撃は外れた	←	サニーゴのころがる
		毒ダメージ HP86→46
ターン6		
ヨルノズクのそらをとぶ	→	サニーゴ HP46→30
ヨルノズク HP127→85	←	サニーゴのころがる
		毒ダメージ HP30→0
		ニョロトノを繰り出した

猛毒状態の相手に対しそらをとぶを使い、ターン稼ぎ。

ころがるは継続するごとに威力が増加するが、そらをとぶによりかわされて失敗。

猛毒のダメージで倒しきる。

▲図 3.6　バトルログ (1/2)

ターン7		
ヨルノズク HP85→15	←	ニョロトノのれいとうビーム
ヨルノズクのどくどく	→	もうどくをあびた
ターン8		
ヨルノズク HP15→0	←	ニョロトノのれいとうビーム
サンドを繰り出した		
ターン9		
		ハリーセンを繰り出した
サンドのじしん	→	ハリーセン HP171→27
ターン10		
スリーパーを繰り出した		
スリーパー HP191→150	←	ハリーセンのなみのり
ターン11		
スリーパー HP150→82	←	ハリーセンのヘドロばくだん
スリーパーのサイコキネシス	→	ハリーセン HP27→0
		ニョロトノを繰り出した
ターン12		
スリーパー HP82→72	←	ニョロトノのれいとうビーム
		毒ダメージ HP215→189
スリーパーのれいとうパンチ	→	ニョロトノ HP189→178
ターン13		
スリーパー HP72→20	←	ニョロトノのおんがえし
		毒ダメージ HP178→152
スリーパーのサイコキネシス	→	ニョロトノ HP152→107
ターン14		
スリーパー HP20→0	←	ニョロトノのれいとうビーム
サンドを繰り出した		
ターン15		
サンド HP171→27	←	ニョロトノのれいとうビーム
		毒ダメージ HP107→81
サンドのじしん	→	ニョロトノ HP81→11
たべのこし HP27→37		
ターン16		
サンド HP37→0	←	ニョロトノのれいとうビーム
バトル終了 勝者 パーティ2		

毒を浴びてない相手にはどくどくを使用。

素直にれいとうビームを使用しなかったのは疑問手。

こうかはいまひとつ。学習不足。

▲図 3.7　バトルログ (2/2)

第 4 章

まとめ

本巻では、ポケモン金銀ルールでのポケモンバトルにおける戦略を AI に考えさせました。

AI を学習させるために必要なポケモンバトルのシミュレータを初代ルールにのみ対応している自作のものからオープンソースのシミュレータに変更し、それに伴いソフトウェア実装を刷新しました。金銀ルールで新たに導入された要素である持ち物を活用するためパーティ生成におけるパーティ評価関数に入力する特徴量を拡張しました。実験の結果、「たべのこし」が有効な道具であることが自動的に発見されました。バトル中の行動方策の学習では、各行動の確率を直接的にモデリングする Actor-Critic 型の強化学習を用いました。あえてモデルを線形モデルとし、ブラックボックスになりがちな AI の思考パターンを可視化することを試みました。最も強かったパーティは攻撃技ばかりを使うものでしたが、「どくどく」を使った後「そらをとぶ」でターン稼ぎをして毒のダメージで相手を倒すという戦略をとったパーティが存在しました。

本巻で金銀ルールの第一歩を踏み出すことはできましたが、まだまだ人が見て賢いと思える域には全く至っていません。次のステップとしては行動方策の学習をよりうまく進むようにして明らかな悪手（こうかはいまひとつの技を用いるなど）を減らしていくことが必要と思われました。パーティ生成面では、特定の組み合わせで有効となる技のコンボをどうやって発見するかや、ランダムに使用するだけではデメリットしかない「だいばくはつ」と単に無意味な「はねる」の違いをどう見分けて生かすかということが課題になります。

付録 A

金銀ルールにおける持ち物一覧

金銀ルールでは、後継となるルビー・サファイア（第 3 世代）以降には存在しない持ち物が多数あります。直感的な名称のものも多いですが、ポケモン金銀をプレイされていない方のために持ち物と効果の一覧を掲載しておきます。きのみについては名称が変更されただけで第 3 世代に同等のアイテムが存在します。この世代に特有なのは「はかいのいでんし」です。

▼表 A.1　特定タイプの技の威力を 1.1 倍にする持ち物

名前	対応するタイプ
ピンクのリボン	ノーマル
みずたまリボン	ノーマル
しんぴのしずく	みず
きせきのタネ	くさ
もくたん	ほのお
じしゃく	でんき
とけないこおり	こおり
くろおび	かくとう
くろいメガネ	あく
どくバリ	どく
やわらかいすな	じめん
するどいくちばし	ひこう
まがったスプーン	エスパー
かたいいし	いわ
ぎんのこな	むし
のろいのおふだ	ゴースト
りゅうのキバ	ドラゴン
メタルコート	はがね

▼表 A.2　きのみ系の持ち物（一度発動すると消滅）

名前	効果
きのみ	HP を 10 回復（最大 HP の 1/2 未満になった時）
おうごんのみ	HP を 30 回復（最大 HP の 1/2 未満になった時）
にがいきのみ	こんらん状態を回復
やけたきのみ	こおり状態を回復
こおったきのみ	やけど状態を回復
はっかのみ	ねむり状態を回復
まひなおしのみ	まひ状態を回復
どくけしのみ	どく状態を回復
きせきのみ	すべての状態異常を回復
ふしぎなきのみ	PP が 0 になった技の PP を 5 回復

▼表 A.3　その他の持ち物（*は一度発動すると消滅）

名前	効果
たべのこし	毎ターン終了時に最大 HP の 1/16 回復
きのみジュース*	HP を 20 回復（最大 HP の 1/2 未満になった時）
はかいのいでんし*	登場時に攻撃 2 段階アップし、混乱する
でんきだま	ピカチュウに持たせると、ダメージ計算時に特攻を 2 倍にする
ラッキーパンチ	ラッキーに持たせると、急所率を 25% にする
ふといホネ	カラカラ・ガラガラに持たせると、ダメージ計算時に攻撃を 2 倍にする
メタルパウダー	メタモンに持たせると、ダメージ計算時に防御・特防を 2 倍にする
ながねぎ	カモネギに持たせると、急所率を 25% にする
せんせいのツメ	毎ターン 23.4% の確率で先制攻撃できる
おうじゃのしるし	11.7% の確率で、攻撃時に相手をひるませる
ピントレンズ	急所ランクを +1 する
ひかりのこな	相手の技の命中率を-7.8% する
きあいのハチマキ	11.7% の確率で、通常ならひんしになるダメージを受けても HP が 1 残る

なお Pokémon-Showdown のデータベースからアイテム一覧をとってきた際、次の道具が混入しています。ちからのこな・たいようのいし・メール・アップグレード。これらは持たせても効果がありません。

付録 B

リトルカップでのパーティ評価関数学習

AI による戦略生成の面白い点のひとつとして、ルールを変更して計算を行えば新しいルールに対応した戦略が得られる点があります。本巻ではシミュレータが自作のものではなくなってしまったため自由度は下がりましたが、パーティ生成のルールだけ書き換えれば実験できる「リトルカップ」を実装し、パーティ評価関数を学習してみました。

リトルカップは、参加できるポケモンや技に制約のあるルールです。ルールをポケモン Wiki[*1] より引用します。

> 以下の条件をすべて満たしたポケモンのみが出場できる。
> ・レベル 5 以下　・タマゴからうまれることができる。　・その種類のポケモンが別のポケモンに進化できる。　・その種類のポケモンに進化するポケモンがいない (2 進化ポケモンの 1 進化形態では出場不可能)。　・同じ種類のポケモンは 1 匹のみ。　・同じどうぐを 2 匹以上に持たせることはできない。
> また、バトル中のルールは以下のとおりである。
> ・6 匹の中からシングルバトルは 3 匹、ダブルバトルは 4 匹を選んで対戦する。　・2 匹以上同時にねむり・こおりにはできない。　・じばく・だいばくはつでお互いのポケモンがすべてひんしになったらそのわざを出した側の負け。　・最後のポケモンのほろびのうた・みちづれは必ず失敗する。　・りゅうのいかり・ソニックブームは必ず失敗する (ポケモンの最大 HP が低く、強すぎる技となってしまうため)。

[*1] https://wiki.xn--rckteqa2e.com/wiki/%E3%83%AA%E3%83%88%E3%83%AB%E3%83%90%E3%83%88%E3%83%AB

ルールを簡単に説明すれば、低レベルかつ進化前の弱いポケモンでどう戦うのかが問われるルールということになります。出場できるポケモン数は 86 種類となります。なお今回の実装では、「りゅうのいかり・ソニックブームは必ず失敗する」ではなく、これらの技をパーティ生成から除外することとしました。

パーティ評価関数の学習方法は本編と同じです。学習した結果として得られた係数を図 B.1・図 B.2 に示します。P 特徴を見ると、ゴースが他を引き離してトップの数値です。技マシンで強力な技を多数覚えられるからでしょうか？ 次点でラッキーです。ラッキーは第 2 世代で新たに進化先であるハピナスが追加されたため、リトルカップに参加できるようになりました。もともと進化しないポケモンとして実装されたため比較的強力です。ストライク、イワークも同様の事情でしょう。ピチュー、ピィ、ププリンなど既存のポケモンの進化前として新規実装されたポケモン（いわゆるベイビィポケモン）は軒並み低評価でした。M 特徴では、本編と同様ころがるが最強技でした。進化前ポケモンは原則はかいこうせんを覚えられないので、それがリストに上がっていない以外はあまり変わらない感じでしょうか。I 特徴では、たべのこしよりもおうごんのみ、きのみジュースが上位を占めています。それぞれ HP を 30、20 回復する持ち物であり、レベル 5 という HP の絶対値が小さい環境でより効果的というのは理解しやすい結果です。2 要素複合の特徴はやはりあまりわかりやすい結果にはなっていません。

▲表 B.1　パーティ評価関数を最大にするよう最適化したパーティ

ポケモン	LV	持ち物	技1	技2	技3	技4
ムチュール	5	たべのこし	れいとうビーム	れいとうパンチ	ふぶき	サイコキネシス
ウパー	5	おうごんのみ	おんがえし	なみのり	ずつき	じしん
ラッキー	5	きのみジュース	かみなり	ころがる	バブルこうせん	１０まんボルト
ヒメグマ	5	きのみジュース	じしん	ずつき	れいとうパンチ	ころがる
ブビィ	5	おうごんのみ	かえんほうしゃ	クロスチョップ	サイコキネシス	かみなりパンチ
デルビル	5	きのみ	かみくだく	かみつく	おんがえし	だいもんじ
コイル	5	おうごんのみ	ころがる	かみなり	かげぶんしん	１０まんボルト
ドードー	5	きのみジュース	のしかかり	ドリルくちばし	おんがえし	トライアタック
オムナイト	5	たべのこし	なみのり	バブルこうせん	れいとうビーム	ハイドロポンプ
エレキッド	5	おうごんのみ	かみなり	１０まんボルト	かみなりパンチ	スピードスター
ケーシィ	5	たべのこし	サイコキネシス	れいとうパンチ	でんじほう	ほのおのパンチ
テッポウオ	5	きのみジュース	バブルこうせん	れいとうビーム	おんがえし	なみのり
イシツブテ	5	おうごんのみ	どろかけ	かいりき	かえんほうしゃ	ずつき
ウパー	5	きのみ	ころがる	あなをほる	じしん	なみのり
ゴース	5	きのみジュース	サイコキネシス	おんがえし	１０まんボルト	ギガドレイン

このパーティ評価関数を最大化するパーティを本編と同様の手法で生成してみました。表 B.1 に示します。見事にフルアタ構成（攻撃技のみが含まれているという意味）となっています。特にムチュールはれいとうビーム・れいとうパンチ・ふぶきという同系統の技が 3 つもあり、バラエティ不足が厳しいです。進化前かつレベル 5 というただでさえ覚える技の範囲が狭い状況なので、ランダムに行動する前提ではこうなってしまうのも仕方ありません。

P特徴	係数
ゴース	0.248
ラッキー	0.190
ドードー	0.172
コイル	0.170
ストライク	0.163
カブト	0.154
オムナイト	0.153
イワーク	0.135
サイホーン	0.130
ポニータ	0.101
…	
ハネッコ	-0.096
ププリン	-0.100
マグマッグ	-0.104
トゲピー	-0.104
レディバ	-0.110
キャタピー	-0.113
イトマル	-0.120
ピィ	-0.123
ヒマナッツ	-0.143
ピチュー	-0.159

M特徴	係数
ころがる	0.349
なみのり	0.332
じしん	0.316
ハイドロポンプ	0.285
サイコキネシス	0.273
たきのぼり	0.256
１０まんボルト	0.246
ヘドロばくだん	0.246
れいとうビーム	0.238
かえんほうしゃ	0.237
…	
ねごと	-0.176
テレポート	-0.176
くろいきり	-0.177
いとをはく	-0.178
あまいかおり	-0.181
はらだいこ	-0.184
みやぶる	-0.187
じこあんじ	-0.192
きあいだめ	-0.201
がまん	-0.202

I特徴	係数
おうごんのみ	0.276
きのみジュース	0.255
たべのこし	0.190
きのみ	0.168
きせきのみ	0.071
ひかりのこな	0.037
おうじゃのしるし	0.029
どくけしのみ	0.024
にがいきのみ	0.014
きあいのハチマキ	0.010
…	
でんきだま	-0.028
たいようのいし	-0.029
メタルコート	-0.030
とけないこおり	-0.033
やけたきのみ	-0.034
ちからのこな	-0.036
するどいくちばし	-0.037
のろいのおふだ	-0.038
りゅうのキバ	-0.049
はかいのいでんし	-0.292

PP特徴	係数
エレキッド・オムナイト	0.066
サンド・タマタマ	0.066
キャタピー・ホーホー	0.065
ププリン・ゴース	0.064
イワーク・エレキッド	0.058
フシギダネ・ヤドン	0.055
ハネッコ・ゴマゾウ	0.054
ドードー・ネイティ	0.054
ゴース・オムナイト	0.053
コラッタ・クラブ	0.053
…	
ポッポ・メリープ	-0.054
ベトベター・スリープ	-0.056
シェルダー・イトマル	-0.057
ニョロモ・ハネッコ	-0.057
ブビィ・マグマッグ	-0.057
ヒノアラシ・マグマッグ	-0.059
ニャース・メリープ	-0.059
ピチュー・マグマッグ	-0.061
ネイティ・ウパー	-0.063
イトマル・クヌギダマ	-0.075

MM特徴	係数
かげぶんしん・かみなり	0.121
かげぶんしん・ころがる	0.107
のろい・すてみタックル	0.101
とっしん・みずでっぽう	0.093
あなをほる・アイアンテール	0.093
かげぶんしん・ソーラービーム	0.091
フラッシュ・たいあたり	0.090
すてみタックル・ねむる	0.090
まるくなる・ころがる	0.089
まもる・ころがる	0.086
…	
がまん・まもる	-0.103
こらえる・ねごと	-0.105
メロメロ・にほんばれ	-0.112
まもる・ねごと	-0.112
いとをはく・たいあたり	-0.113
メロメロ・かげぶんしん	-0.116
こらえる・にほんばれ	-0.121
ねむる・こうごうせい	-0.124
まもる・あまごい	-0.125
メロメロ・のろい	-0.138

▲図 B.1　パーティ評価関数の係数 (1/2)

PM特徴	係数
カブト・ころがる	0.179
ケーシィ・サイコキネシス	0.151
デルビル・かみくだく	0.142
ドードー・ドリルくちばし	0.141
コイル・１０まんボルト	0.122
チョンチー・たきのぼり	0.121
ムチュール・れいとうパンチ	0.120
ムチュール・こごえるかぜ	0.115
コイル・ころがる	0.113
トゲピー・プレゼント	0.113
…	
チコリータ・メロメロ	-0.086
コラッタ・１０まんボルト	-0.088
ウリムー・しろいきり	-0.096
カブト・のろい	-0.097
スリープ・でんじは	-0.101
イトマル・クモのす	-0.107
ヒマナッツ・ヘドロばくだん	-0.108
キャタピー・たいあたり	-0.113
キャタピー・いとをはく	-0.113
ピチュー・プレゼント	-0.133

PI特徴	係数
ズバット・のろいのおふだ	0.081
イーブイ・きのみジュース	0.076
ヒマナッツ・たいようのいし	0.068
コラッタ・どくけしのみ	0.058
ウリムー・おうじゃのしるし	0.057
ストライク・きせきのみ	0.053
ニャース・ちからのこな	0.051
ミニリュウ・きのみジュース	0.051
カラカラ・くろおび	0.050
ナゾノクサ・どくバリ	0.050
…	
メリープ・まがったスプーン	-0.047
ガーディ・くろおび	-0.048
ヨーギラス・たべのこし	-0.049
ロコン・こおったきのみ	-0.050
ドガース・するどいくちばし	-0.052
ズバット・じしゃく	-0.054
チコリータ・まがったスプーン	-0.055
カラカラ・きせきのタネ	-0.055
ハネッコ・たいようのいし	-0.055
ドガース・やわらかいすな	-0.059

MI特徴	係数
おんがえし・きのみジュース	0.099
かいりき・おうごんのみ	0.095
すてみタックル・きのみジュース	0.095
のしかかり・にがいきのみ	0.089
アイアンテール・どくけしのみ	0.085
のしかかり・きのみジュース	0.083
れいとうビーム・おうごんのみ	0.080
すてみタックル・おうごんのみ	0.079
おんがえし・まがったスプーン	0.077
れいとうパンチ・せんせいのツメ	0.077
…	
がまん・しんぴのしずく	-0.083
ねむる・きのみジュース	-0.085
じこあんじ・とけないこおり	-0.086
こらえる・せんせいのツメ	-0.088
みがわり・たいようのいし	-0.089
ねごと・しんぴのしずく	-0.090
アイアンテール・ながねぎ	-0.092
いびき・ひかりのこな	-0.095
ねごと・きせきのみ	-0.098
メロメロ・きせきのタネ	-0.106

▲図 B.2　パーティ評価関数の係数 (2/2)

あとがき

2017 年 12 月に準備編と称した PDF を公開してから 2 年が経過しました。ついにポケモン金銀編に突入です。初代ポケモンを研究しつくしたというわけではもちろんなく、むしろ現在の残念な AI でも攻撃技を連打する以外の戦略を発見しやすいであろう環境としてポケモン金銀へ移行したという事情です。バトルには関係ないため本文には記載しませんでしたが、ポケモン金銀はいろんな点で革新的でした。ゲームカートリッジ内に電池と時計が内蔵され、時間・曜日の概念が追加されました。夜にしか出ないポケモン、月曜日朝にしか売っていない道具など、わくわくする仕掛けがたくさんありました。新しい地方であるジョウト地方だけでなく、初代の地方であるカントー地方にも行けてバッチが 16 個取れたり、初代と通信交換ができたりと、世界が広がったということが強く感じられました。マイナーチェンジ版のクリスタルでは、携帯電話と接続してネット対戦ができるという先進的すぎる機能もありました。筆者は当時携帯電話を持っておらず試せませんでしたが。

思い出深いポケモン金銀ですが、AI 実装は難航しました。シミュレータの仕様を読み解いてインターフェースを作るという部分に時間がかかり、アルゴリズムの実装が開始できないまま 11 月中旬に至ってしまい追い詰められましたが、なんとか過去のアルゴリズムを実装・拡張して最低限の戦略を観察することができました。今回が初めてのコミックマーケットへのサークル参加ですが、読者に満足いただける本になっていれば幸いです。

実験の過程で新しいアイデアもいくつか思いついているので、今後実装していきたいと思います。また、このプロジェクトで用いている深層学習フレームワークの Chainer（これをベースに強化学習機能を付加するのが ChainerRL）ですが、執筆中の 2019 年 12 月 5 日に開発終了宣言がなされました。今までお世話になりました。フレームワークの移行もしないといけませんね。ただひとまずは、最新作のポケモンソード・シールドで遊びたいですね。

本書の電子版を `https://select766.booth.pm/items/1722869` から無料ダウンロードできます。BOOTH アカウントが必要。暗号化 zip の解凍パスワードは `wnhz9vV66i` です。

ＰｏｋéＡＩ　＃３：金銀導入編

2019 年 12 月 31 日　コミックマーケット９７ v1.3.0

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

#3.5: 金銀導入編追加 DLC

このパートは、コミックマーケット 97（2019 年 12 月）発行の第 3 巻の執筆から技術書典 8（2020 年 3 月）までの間の開発の進捗を本と同様の形式でまとめたものです。ブログ記事をまとめた上で図表を追加することで、速報性は下がるものの一貫性を増し読みやすい内容にすることを目指しています。Web 小説を加筆修正して挿絵をつけて出版するような感じでしょうか？ 出版業界で働いたことはないので想像ですが…。第 3 巻購入者への電子版限定追加コンテンツ（DLC: Download Contents）として配布し、今後第 4 巻の一部となる予定です。なお、技術書典 8 は新型コロナウイルス問題により中止となったため、その代替となるオンラインイベント「技術書典　応援祭」での発表となります。

目次

第 1 章

汎用行動選択モデルの学習

1.1　イントロダクション

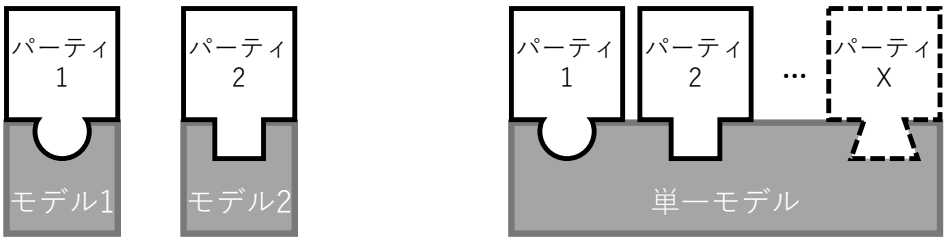
今までポケモンバトル中の行動選択を行うモデルは、パーティごとに別個のパラメータを学習していました。しかし今後の発展を考えるとこの方式は難点があり、あらゆるパーティの行動選択を行える単一のモデルを学習することを考えます。

従来の行動選択モデルは、パーティごとに別個のパラメータを学習します。モデルの入力はバトル中の状態で、相手のポケモンの種族、自分と相手それぞれの HP の残り比率、ランク補正状態などです。モデルの出力は技 1〜4 それぞれの優先度を表す値となります。ここで優先度とは、実数値で大きいほどその技を選択すべきであることを表します[*1]。今回の議論ではポケモン交代は本質的ではないため、ポケモン 1 匹だけを出す 1vs1 バトルで考えます。モデルの入力として、自分のポケモンの種族やどの技を覚えているかという情報はありません。技 1〜4 が何であるか知らずとも、水タイプの相手に最大ダメージを与えられるのは技 1 で、炎タイプの相手に最大ダメージを与えられるのは技 3 であるというような情報を勝敗から学習していけば十分であるためです。この方式の問題点は、パーティ構成が変われば一から学習しなければならないという点です。プロジェクトの大きな目標として、パーティ構成とバトル中の行動選択の両方を最適化したいと考えています。さまざまなパーティを試行錯誤して作る際、過去に学習した行動選択モデルの情報が使えれば、学習コストを下げられる可能性があります。自分のパーティ構成がなんであろうと、水タイプの相手には 10 まんボルトやメガドレインが有効である、というような知識は有用です。そのような知識をどのパーティでも共有し、新しく生成されたパーティでも、相手が水タイプで 10 まんボルトとれいとうビームを覚えているなら 10 まんボルトを

[*1] でんこうせっかが素早さに関係なく先制できるという意味での優先度ではないことにご注意ください。より誤解の少ない日本語があるといいのですが…

選ぶということを行動選択モデルの学習なしに実現できれば、さまざまなパーティの強さを（ランダムに行動させるより正確に）評価することが容易になります。パーティの技 1〜4 の効果が固定という前提が取り払われれば、別のメリットとして、「へんしん」等で自分の技が変化した場合にも対応することができるようになります。ただし、タイプ相性すら難儀している現状では難しすぎる課題のため当分は取り組みません。

これを実現するために、自分のパーティにかかわらず 1 つのモデルパラメータを共有する「汎用行動選択モデル」を学習したいと思います。これに対して従来の行動選択モデルは「パーティ固有行動選択モデル」と呼ぶことにします。汎用行動選択モデルでは、入力として自分のパーティのポケモンの種族と覚えている技も与えて、ターンごとにどの技を使うかを出力します。



▲図 1.1　パーティ固有行動選択モデルと汎用行動選択モデルの比較。

パーティ固有行動選択モデルと汎用行動選択モデルの違いを表す模式図を図 1.1 に示します。パーティ固有行動選択モデルでは、パーティが決まった上でそれを運用するためのパラメータを一から学習します。異なるパーティに対応するモデルは情報を何も共有していません。汎用行動選択モデルは、自由度の高い（学習対象となるパラメータ数の多い）モデル 1 つを学習し、多数のパーティに対応します。対応するパーティは学習時に出現した物だけでなく、未知のパーティも含みます。たとえば学習時のパーティとして「ハピナス＋どくどく＋かげぶんしん」および「ブラッキー＋どくどく＋かげぶんしん」がいて、いずれもどくどくを使ったあとにかげぶんしんを使うという戦略が有効であれば、学習時には出現しなかった「エアームド＋どくどく＋かげぶんしん」でも同様の戦略が有効かもしれない、と推測できます。汎用行動選択モデルでは、類似したパーティでは類似した戦略で行動させることが可能となります。もちろん、学習データに一度も出現していないポケモンや技があればうまくいきません。システム上、技はただの番号で管理されます。た

とえばわざマシン 01 が「ばくれつパンチ」で 02 が「ずつき」のときに 03「のろい」の効果を予想する、ということはできません。あくまで見たことのあるポケモンや技について、新しい組み合わせでも既存の知識が活用できるよう、機械学習により汎化能力をもたらすことが目標です。また、耐久力が低いポケモンであるマルマインではどくどく＋かげぶんしんという戦略はハピナスほど効果がない、ということを考慮するには同じ技であっても耐久力の高低、相手とのタイプ相性などで有効性がどう変化するかが学習される必要があります。どれだけ多様な組み合わせに対応できるかは学習データの分量やモデル設計の良し悪しで決まります。

パーティ固有行動選択モデルではあくまで自分が覚えている技 4 つのどれを選択するかだけを考慮すればよかった一方、汎用行動選択モデルでは 100 種類以上存在する技・ポケモンすべての運用を知っている必要があるため、学習すべきパラメータ数が増加します。また、どのようなモデル構造（ニューラルネットワークの構造）にすればいいかも明らかではありません。パーティ固有行動選択モデルでも勝敗から強化学習した結果のモデルが「こうかはいまひとつ」な技を選択してしまうなど難ありの状況のため、パラメータ数が増加したモデルの強化学習は難しいと予想しました。この課題を解決するアイデアとして、強化学習よりも容易である教師あり学習で汎用行動選択モデルを学習させ、どのようなモデル構造・パラメータ数の規模ならうまく学習できるのかを検討することにしました。教師あり学習には教師データ、すなわちバトルの状況と適切な行動（技）の組が必要です。しかしそのようなデータはどこかで拾えるわけでもないので自作する必要があります。1 つの方法は人力で作成することですが、数千件必要であろうサンプルを作るのが大変そうだし条件が変わるたびに作り直しになるので不採用です。そこで今回は、パーティ固有行動選択モデルを多数のパーティに対して学習させ、モデル間で対戦させてバトルの状態とモデルが選んだ行動の組を収集し、疑似的な教師データとして用いることにしました。ゲーム AI としてこのような手段を用いた事例があるかは分かりませんが、10 万種類の画像分類を行うモデルを学習するため、自動車ジャンルだけ、鳥ジャンルだけといった特化モデルを学習し、その知識を 1 つの汎用モデルに吸収させる研究があります[*2]。

今回の研究の流れは次のようになることを想定しています。

- 全ポケモン・技を候補にするとパーティ固有行動選択モデルの必要数が非常に多くなるため、適宜削減した環境を作成する
- ランダムに生成したパーティに対してパーティ固有行動選択モデルを学習させる
- 学習したモデルを対戦させて、バトルの状態（自分のパーティの情報を含む）と選択した行動のペアを収集し教師データとする

[*2] Jiyang Gao et al., Knowledge Concentration: Learning 100K Object Classifiers in a Single CNN. arXiv 1711.07607.

- 汎用行動選択モデルの構造について教師あり学習で試行錯誤する
- 汎用行動選択モデルを直接的に強化学習させる（未実現）

1.2 学習環境の制約

第 3 巻では特殊なものを除いてポケモン金銀のすべてのポケモン・技をパーティに組み込める条件としましたが、自由度を少しでも減らして学習を容易にするためポケモン・技の数を制限した環境を用いることとしました。本節ではその手法を説明します。

バトルのルールの前提条件はこのように設定しました。

- ポケモン金銀ルール
- パーティのポケモンは 1 匹のみ（1vs1）で、LV55

3vs3 は選択肢・パーティのもつ情報量が多くなり学習が難しくなることが予想されるので、まずは一番簡単な条件に設定しました。

従来の行動学習で扱いづらかった点として、ポケモンや技の強さに格差が大きい点がありました。たとえばトランセルはどういう戦略をとろうが勝ち目はほとんどなく、「なみのり・みずでっぽう・あわ・しっぽをふる」を覚えたカメックスではバトル中の状況にかかわらずなみのりを選択するのが最適解となるでしょう。このような状況下の行動選択データは無意味なデータとなってしまう可能性が高く、モデルの学習の妨げになると考えられます。また、トランセルの行動選択モデルの学習は純粋に時間の無駄になります。そこで、ある程度有効なポケモン・技だけを抽出することで無意味な解に陥らない環境を作成します。

まずポケモンですが、最終進化系だけを用いることにしました。ただし、禁止級伝説、技マシンが使えないポケモンを除外しています。すなわちミュウ、ミュウツー、ホウオウ、ルギア、セレビィ、メタモン、アンノーン、ソーナンス、ドーブル以外の最終進化系で、合計 129 種類となりました。他の手法として、種族値で決める方法もあるかと思います。

次に技ですが、ポケモンほど自明な基準がありません。攻撃技なら威力という基準がありますが、補助技は比較軸がありません。そこで、ランダムな技を覚えたパーティを多数生成し、第 2・3 巻で述べたパーティ評価関数の手法で技の有用度を定量化することとしました。

ポケモンは最終進化系だけをランダムに選択し、技はそれぞれのポケモンが覚えられるものからランダムに 4 つ選択します。このようなパーティを 10,000 個生成し、ランダムに行動させた勝敗から各パーティのレートを計算します。ここでは技がレートに与える影響を抽出したいので、パーティ特徴量として技（M）特徴量のみを用います。

技ごとに算出された係数の最高 10、最低 10 を表示します。係数が大きいほど勝利に貢献していることを表します。

▼表 1.1　技ごとのパーティ評価関数の係数

技	係数
はかいこうせん	0.660
じしん	0.561
なみのり	0.493
１０まんボルト	0.477
れいとうビーム	0.460
のしかかり	0.418
かいりき	0.415
ハイドロポンプ	0.406
かえんほうしゃ	0.393
ちきゅうなげ	0.392
…	…
あまごい	-0.411
やつあたり	-0.411
いびき	-0.423
ねごと	-0.449
こらえる	-0.456
しんぴのまもり	-0.479
メロメロ	-0.484
だいばくはつ	-0.490
じばく	-0.760
いとをはく	-0.862

はかいこうせん、じしんなどの強そうな技が上位に来ています。下位では、1vs1 バトルなので使うと即負けになるじばくのほか、使用条件を満たさないと無意味なメロメロ、ねごとなどが出ています。これらの技の有効な運用もいつかは実現したいですが、「こうかはばつぐん」の技を選ぶことすら苦労している現状なので後回しです。

ポケモンごとに技構成の自由度をもたせつつ無意味な技を排除するため、各ポケモンの覚えられる技のうち係数上位 8 個を選択し、全（最終進化系）ポケモンにわたって和集合をとりました。その結果、次の 52 個の技が選択されました。

▼リスト 1.1　パーティ評価関数をもとに選択された技リスト

つのでつく，ふぶき，つばさでうつ，どくどく，やどりぎのタネ，ふみつけ，いあいぎり，はかいこうせん，でんじほう，サイコキネシス，かいりき，ソーラービーム，ロケットずつき，げんしのちから，そらをとぶ，ナイトヘッド，おんがえし，はなびらのまい，なみのり，かみなりパンチ，たきのぼり，かげぶんしん，いわくだき，どくのこな，とっしん，つのドリル，どろかけ，のしかかり，ゴッドバード，ころがる，スピードスター，れいとうビーム，ちきゅうなげ，ヘドロばくだん，すてみタックル，バブルこうせん，ほのおのパンチ，だいもんじ，ハイドロポンプ，サイケこうせん，かみなり，はっぱカッター，じしん，ドリルくちばし，ばくれつパンチ，ずつき，はがねのつばさ，すなあらし，れいとうパンチ，１０まんボルト，ギガドレイン，かえんほうしゃ

以上、ルール上存在するポケモン・技のうち、ランダムに選択しても大きく格差が出ないよう有効なポケモン 129 種類・技 52 種類からなるサブセットを抽出しました。この範囲で汎用行動選択モデルの学習を進めていきます。

まったく別の手段として、人間同士の公式大会（ニンテンドウカップ）で使われたポケモン・技に絞るというやり方もあるかと思います。人間の思考に影響されてしまうとはいえ、ランダムに技を使った時の効果で測定するよりは多様な補助技を含められることが期待できます。

1.3 擬似教師データの作成

この節では、モデルの構造を教師あり学習で検討するための疑似教師データの作成を考えます。

汎用行動選択モデルとして Deep Neural Network （DNN）を用いますが、パーティの情報を取り入れるための構造や、パラメータ数の自由度が非常に大きいため適切な規模のものを選ぶ必要があります。モデルを強化学習させると強化学習自体にもさまざまなパラメータが必要（割引率や replay buffer のサイズ等）で、探索が大変です。そのため強化学習の適切なパラメータがわかっているパーティ固有行動選択モデルを用いて局面ごとの行動の正解を生成します。パーティ固有行動選択モデルは 1 パーティに対し 1 つのモデルが対応します。さまざまなパーティに対しそれぞれパーティ固有行動選択モデルを強化学習させ、その行動データを集積し、それを 1 つの汎用行動選択モデルに教師あり学習させるという流れを提案します。パーティ固有行動選択モデルにより選択された行動を疑似教師データと呼ぶことにします。「疑似」とついているのは、パーティ固有行動選択モデルの強化学習が完ぺきではなく、必ずしも最適な行動を選択できるわけではないということを指しています。

疑似教師データの作成は次のように実装しました。パーティをランダムに 1,000 個生成させ、それぞれに対してパーティ固有行動選択モデルを学習させます。パーティの条件は前パートで検討した最終進化系ポケモンだけを含むものです。強化学習中の対戦相手は自分以外のパーティがランダムに行動するエージェントです。こうして学習させたエージェ

ント同士を対戦させ、対戦ログを保存する仕組みを用意しました。対戦ログには、各ターンにおける状況（相手のポケモンの種族、残り HP、状態異常など）・自分のパーティの情報（ポケモンの種族、覚えている技など）・エージェントが選択した行動が含まれます。

実際のログを整形していくつか表示します。

▼リスト 1.2　疑似教師データの凡例

```
自分のポケモン
自分のポケモンの技構成
相手のポケモン
=> 選択した行動
```

▼リスト 1.3　疑似教師データの例

```
自分 マタドガス 187/187
ころがる １０まんボルト でんじほう だいもんじ
相手 オムスター 193/193
=> １０まんボルト

自分 ラフレシア 198/198
はかいこうせん やどりぎのタネ ギガドレイン おんがえし
相手 ウソッキー 193/193
=> ギガドレイン

自分 ラプラス 259/259
いわくだき １０まんボルト サイコキネシス ハイドロポンプ
相手 オクタン 198/198
=> サイコキネシス

自分 ブーバー 187/187
すてみタックル サイコキネシス いわくだき ずつき
相手 ジュゴン 215/215
=> いわくだき
```

必ずしも最適とはいえないですが、ある程度「こうかはばつぐん」な行動を選べています。

次の例は 1 つのバトル中の各ターンの行動を示したものです。

▼リスト 1.4　1 つのバトルから抽出した疑似教師データの例

```
自分 プテラ 204/204
かげぶんしん つばさでうつ どくどく だいもんじ
相手 ポリゴン２ 209/209
=> どくどく

自分 プテラ 157/204
かげぶんしん つばさでうつ どくどく だいもんじ
相手 ポリゴン２ 185/209 tox
```

```
=> だいもんじ

自分 プテラ 108/204
かげぶんしん つばさでうつ どくどく だいもんじ
相手 ポリゴン2 106/209 tox
=> かげぶんしん

自分 プテラ 63/204   evasion+1
かげぶんしん つばさでうつ どくどく だいもんじ
相手 ポリゴン2 56/209 tox
=> かげぶんしん

自分 プテラ 63/204   evasion+2
かげぶんしん つばさでうつ どくどく だいもんじ
相手 ポリゴン2 4/209 tox
=> だいもんじ
```

tox は猛毒状態、evasion+1 は回避率が 1 段階上がった状態を表します。このエージェントはどくどくのあとかげぶんしんで逃げ回るという戦略をとっていることが読み取れます。このようにパーティ構成と相手のポケモンだけで技が決まるわけではなく、相手の状態異常といったバトル中の状態に依存して決めることが学習されている場合もあります。

1.4 汎用行動選択モデルの設計

前節で作成した教師データを用いて、汎用行動選択モデルの学習を試みます。自分のパーティの情報を含むバトルの状態を入力とし、適切な行動（技）を選択するモデルを学習することが目標です。

モデルとして DNN を用います。DNN は畳み込み、リカレントなど構造の自由度が極めて高いですが、今回はもっとも単純に f(バトルの状態, 選択肢 i の情報) => 選択肢 i の優先度 という入出力の fully-connected feedforward network にすることとしました。

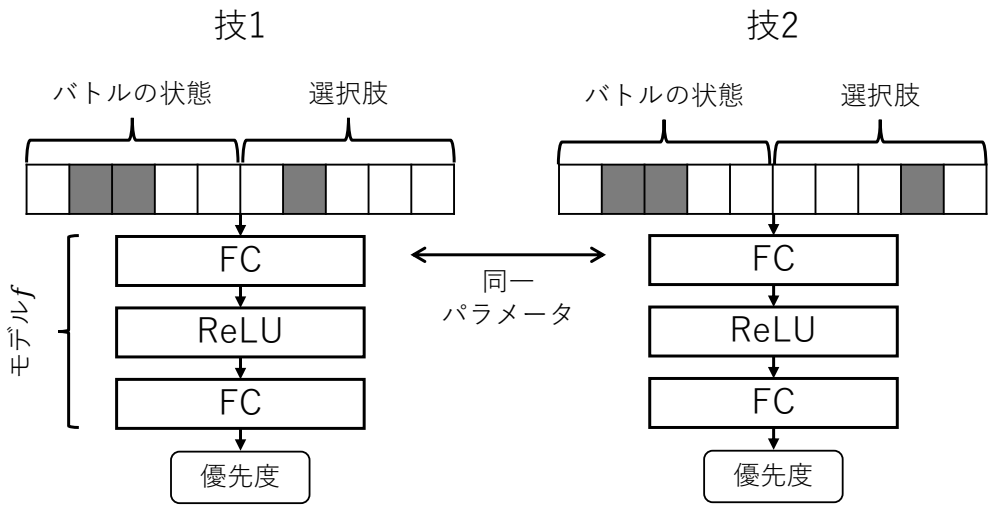
バトルの状態はパーティ固有行動選択モデルで用いたもの[*3]に加え、自分のパーティ構成を与える必要があります。自分のパーティ構成として、ポケモンの種族を与えることとしました。技については選択肢ベクトルに含まれるので、重複を避けてこちらには含めないこととしました。厳密なことをいえば、炎タイプの攻撃技を持っている場合ににほんばれの優先度を上げるというような機能を実現するためには、選択肢の技以外に何を覚えているかを情報として与える必要がありますので、将来的には入力特徴量の情報を増やして表現力を上げることが望ましいです。

[*3] 3vs3 バトルと同等のものを用いていますが、1vs1 のため選択肢は技 4 つだけで、生存ポケモン数などは意味がありません。

▼表 1.2 バトルの状態を表現する特徴量。自分/相手は、どちら側のパーティの情報を入力として与えるか。両方の場合は次元数が倍となる。

特徴	次元数	自分/相手	説明
有効な行動	4	自分	現在どの行動がとれるのかを表す 有効な行動に該当する次元に 1 を設定
生存ポケモン数	1	両方	瀕死でないポケモン数/全ポケモン数
ポケモンタイプ	17	相手	場に出ているポケモンのポケモンのタイプ （ノーマル・水・…）に該当する次元に 1 を設定
HP 残存率	1	両方	場に出ているポケモンの現在 HP/最大 HP
状態異常	6	両方	場に出ているポケモンの状態異常（どく・もうどく・ まひ・やけど・ねむり・こおりのうち 該当次元に 1 を設定)
ランク補正	6	両方	場に出ているポケモンのランク補正（こうげき・ ぼうぎょ・とくこう・とくぼう・すばやさ・命中・ 回避それぞれ、ランク/12+0.5 を設定）
天候	3	-	場の天候（はれ・あめ・すなあらし）に 該当する次元に 1 を設定
ポケモン種族	251	自分	場に出ている自分のポケモンの 種族に該当する次元に 1 を設定

モデルに与える選択肢を表現するベクトルとして今回はもっとも単純に、技を表す 251 次元（技はポケモン数と同じ 251 種類存在）のベクトルを用いました。技の番号に対応する次元に 1 を設定します。実際にパーティに含める技は前述のように 52 種類に絞っているので、出現しない技に対応する次元の値は常に 0 です。バトルの状態ベクトルと選択肢ベクトルを連結したもの、合計 558 次元を DNN への入力とします。



▲図 1.2 モデル・入出力の構成。選択肢（技）ごとに異なる選択肢ベクトルをモデルの入力とし、選択肢の優先度を計算する。

DNN を用いたシステム構成は図 1.2 のようになり、モデル部分の定義をリスト 1.5 に示します。今回から、深層学習ライブラリは PyTorch に移行しました。

▼リスト 1.5 DNN の PyTorch 実装例（fully-connected feedforward neural network、隠れ層 1 層の場合）

```
import torch.nn as nn
import torch.nn.functional as F


class MLPModel(nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, 64)
        self.fc2 = nn.Linear(64, 1)

    def forward(self, x):
        h = x  # batch, feature_dim
        h = F.relu(self.fc1(h))
        h = self.fc2(h)
        return h
```

教師あり学習をするにあたり、一般的な分類問題の定式化に落とし込みます。$softmax(f(\text{バトルの状態, 選択肢 0 のベクトル}), f(\text{バトルの状態, 選択肢 1 のベクトル}), f(\text{バトルの状態, 選択肢 2 のベクトル}), f(\text{バトルの状態, 選択肢 3 のベクトル}))$ を各選択肢の選択確率として、正解データとの cross entropy loss を最小化するように学習します。

先ほど定義したモデルに選択肢ベクトルを変えて 4 回呼び出し、結果を連結して損失を計算することも可能ですが、実装上あまり効率がよくありません。そこで、モデルの呼出しを 1 回ですべての計算を終えるテクニックがあります。それは、全選択肢に対応するベクトルを積み重ねた行列（558 × 4）を入力とし、全結合層をカーネルサイズ 1 の 1D Convolution（畳み込み）に置き換えることです。形式的には畳み込みですが、畳み込みとしての性質はなく一気に全選択肢分のベクトルに対して同一の計算が適用でき、計算結果は等価です。詳細な計算については「1x1 convolution」などで調べてみてください。

▼リスト 1.6　1D Convolution により一気に全選択肢の計算をするコード

```
import torch.nn as nn
import torch.nn.functional as F


class MLPModel(nn.Module):
    def __init__(self, input_dim, n_layers=2, n_channels=64, bn=False):
        super().__init__()
        layers = []
        bn_layers = []
        cur_hidden_ch = input_dim
        for i in range(n_layers):
            layers.append(nn.Conv1d(cur_hidden_ch, n_channels, 1))  # in,out,ksize
            cur_hidden_ch = n_channels
            if bn:
                bn_layers.append(nn.BatchNorm1d(n_channels))
        self.layers = nn.ModuleList(layers)
        self.bn_layers = nn.ModuleList(bn_layers)
        self.output = nn.Conv1d(cur_hidden_ch, 1, 1)
        self.bn = bn

    def forward(self, x):
        h = x  # batch, feature_dim, 4
        for i in range(len(self.layers)):
            h = self.layers[i](h)
            if self.bn:
                h = self.bn_layers[i](h)
            h = F.relu(h)
        h = self.output(h)
        h = h.view(h.shape[0], -1)  # batch, 4
        return h
```

1.5　実験

このように定義したモデルを学習させます。パーティ 1,000 個（それぞれに対するパーティ固有行動選択モデルを学習済み）を 1 パーティ当たり 100 回他のパーティと対戦させ、10 万バトル分の行動を得ました。同じバトルについて 2 つのパーティから見た状態を別のデータとして扱っています。900 パーティ分のデータを学習データ、残り 100 パー

ティ分を評価（validation）データとして使用します。1 回のバトルで複数のターンがあるので、学習データは 34 万サンプルとなりました。

上記のモデルのハイパーパラメータを変更して正解率を測定しました。学習は 10 エポック、Optimizer は Adam、learning rate=0.01、バッチサイズ 256 としました。層数は出力層以外の Convolution の数です。チャンネル数は隠れ層の出力チャンネル数です。バッチ正規化は、各 Convolution の後に学習を安定化させる Batch Normalization レイヤーを付加するか否かです。

▼表 1.3　モデルのハイパーパラメータと正解率

層数	チャンネル数	バッチ正規化	Training 正解率 [%]	Validation 正解率 [%]
1	16	False	55.2	52.5
1	16	True	67.9	55.6
1	64	False	58.2	54.6
1	64	True	72.5	55.2
1	256	False	63.8	53.8
1	256	True	75.1	53.1
2	16	False	65.6	54.2
2	16	True	68.6	56.4
2	64	False	70.5	55.9
2	64	True	73.2	54.6
2	256	False	70.1	55.4
2	256	True	**76.2**	53.2
3	16	False	66.2	**58.5**
3	16	True	67.9	56.0
3	64	False	68.2	57.8
3	64	True	73.6	54.4
3	256	False	68.6	54.8
3	256	True	75.9	53.6

Validation 正解率が最大となるのは 3 層、チャンネル数 16、バッチ正規化なしという結果になりました。ハイパーパラメータ間に極端な差はないですが、層の数は多いほうがよい一方で、チャンネル数が多いと Training 正解率は高くなる一方で Validation 正解率は下がってしまい過学習していることがわかります。バッチ正規化についても、過学習を誘発しているようです。今回の学習データ生成には計算時間が 1 日ほどかかるため、量を大幅に増加させることは難しいです。強化学習に移行するか、過学習しづらいようモデル構造を工夫することが将来課題です。

1.5.1 定性評価

学習した汎用行動選択モデルの挙動を定性的に確認してみます。

前回の記事でもっとも精度が高かったモデル（3 層、チャンネル数 16、バッチ正規化なし）に対し Validation データを入力し、モデルの出力（選択した技）および正解データを表示します。正解データはパーティ固有行動選択モデルが出力したものです。

▼リスト 1.7　凡例

```
自分のポケモン
自分のポケモンの技構成
相手のポケモン
=> 各行動に対する確率
モデル出力=最大確率の技 正解=正解データ
```

▼リスト 1.8　モデル出力の例 1

```
自分 ドククラゲ 204/204
ハイドロポンプ おんがえし とっしん ヘドロばくだん
相手 スターミー 182/182
=> ハイドロポンプ=2.4% おんがえし=7.3% とっしん=0.2% ヘドロばくだん=90.1%
モデル出力=ヘドロばくだん 正解=ヘドロばくだん

自分 ランターン 52/253
なみのり どくどく おんがえし でんじほう
相手 ドンファン 80/215
=> なみのり=89.0% どくどく=3.1% おんがえし=5.4% でんじほう=2.5%
モデル出力=なみのり 正解=なみのり
```

妥当な出力をしています。

▼リスト 1.9　モデル出力の例 2

```
自分 ゴローニャ 138/204 psn
すてみタックル ちきゅうなげ じしん ずつき
相手 パラセクト 25/182
=> すてみタックル=10.7% ちきゅうなげ=18.0% じしん=59.8% ずつき=11.5%
モデル出力=じしん 正解=ずつき
```

じしんはパラセクトに対してタイプ相性が 1/4 なので間違っていると考えられます。

▼リスト 1.10　モデル出力の例 3

```
自分 マリルリ 90/226
れいとうビーム はかいこうせん バブルこうせん どくどく
相手 ラフレシア 11/198
=> れいとうビーム=84.2% はかいこうせん=0.3% バブルこうせん=15.2% どくどく=0.3%
モデル出力=れいとうビーム 正解=どくどく
```

れいとうビームが正しくて、正解データのどくどくは明らかに間違っています。ラフレシアはどくタイプを含んでいてどくどくは効果がありません。正解データも疑似的なものであり必ずしも正しくないため、これに対して正解率 100% を目指すのは得策ではないことを示しています。

モデルの定量的な正解率は 58.5% でしたが、正解データ自体が間違っている場合もあり、定性的には妥当な出力ができているように思われます。

1.6 まとめ

どのパーティに対しても行動選択が行える単一のモデル「汎用行動選択モデル」を学習させる足掛かりとして、既存のパーティ固有行動選択モデルによって生成した行動選択結果を用いて教師あり学習することを試みました。定性的にある程度動作していることが確認できました。かなりパラメータ数の少ないモデルでなければ過学習が起きやすいことがわかりましたので、学習されたモデルの分析を進めたうえでよりよいモデル構造や学習手段の提案を行うことが今後の課題となります。

ＰｏｋéＡＩ　＃３．５：金銀導入編追加ＤＬＣ

2020 年 3 月 7 日　技術書典 応援祭 v1.3.5

著　者　select766 (select766@outlook.jp)
発行所　ヤマブキ計算所
印刷所　（電子書籍配布専用）