PokéAI

～人工知能の考えた最強のポケモン対戦戦略～

[#2：初代 3vs3 編]

- パーティ評価関数の学習に基づく
  パーティ構成の最適化
- 深層強化学習による行動選択
- 付録：初代で「さかさバトル」

@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 2018 年 10 月発行の第 1 巻の続編となります。第 1 巻では、プロジェクトの大きな枠組みを述べるとともに、もっとも単純な条件として 1vs1 のバトルを行いました。ポケモンを 1 匹だけ出すという条件はポケモンバトルとして特殊で、ほとんどポケモン同士の相性で勝敗が決まってしまうような設定だといえます。この第 2 巻では、3vs3 のバトルをターゲットにします。ポケモンバトルの標準的な遊び方に近いものになります。3vs3 バトルを行うには、1vs1 と比べて難しい点があります。1 点目はパーティ構成の自由度が高まる点です。3 匹分のポケモンの種類と技の組み合わせを決めなければならないので、1 匹だけ決める場合より大幅に選択肢が増えます。2 点目はバトル中の行動選択の選択肢が増える点です。1 匹だけであれば、4 つの技のどれかを選ぶだけでしたが、交換という選択肢が追加されること、また場に出ているポケモンによって覚えている技が違うという状況に対処しなければなりません。これらの課題に対して解決策を見出すのが本書の目的となります。

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。

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。例えば、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。

1.1　ポケモンバトルのルールと新たな課題

本書で扱うポケモンバトルのルールを述べます。初代ポケモン（赤・緑）に準拠し、プレイヤー一人が 3 匹のポケモンを持つ 3vs3 バトルを対象とします。トリプルバトルではなく、あくまで 1 プレイヤーあたり 1 匹だけが同時に場に出ます。バトル前にパーティを見せあうことはしません（6 匹の候補をお互いに見せたうえで、3 匹を選ぶ段階を含めるルールのことです。これは将来課題とします）。レベルは 50〜55 で、3 匹の合計値が 155

になるようにします[*1]。強力な状態異常として眠り状態[*2]や氷状態がありますが、複数のポケモンをこれらの状態にすることは禁止していません（複数催眠は禁止でない）。当時の大会ルールでは禁止されていましたが、今回開発した AI の範囲ではこれらを禁止しなくてもゲームバランス崩壊という状況には至っていません。なお、「ふぶき」で相手を凍らせる確率は 10% としました。赤・緑では 30% ですが、さすがに何も考えずにふぶきを連打するのが強すぎて戦略性に欠けるため、ポケモンスタジアム準拠で 10% に変更しました。この設定変更の影響を調べたので、付録をご覧ください。

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[*3]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。

第 1 巻では 1vs1 バトルを対象にしていました。パーティ構成ではポケモン 1 匹の種類と技構成だけを決め、バトル中の行動は技 4 種類の中から 1 つを選ぶだけでした。今回 3vs3 となり、2 点の大きな変更が生じます。

まずパーティ構成ですが、3 匹の種類と技構成が必要です。1 匹のパラメータがポケモンの種類 151 種類と技 30 種類（技は全部で 165 種類ですが、ポケモンごとに覚えることができる技に制限があります）を 4 つで $151 \times 30^4 \simeq 10^8$（1 億）とすると、3 匹だとこれの 3 乗で 10^{24} の組み合わせ候補があります。1 億程度であれば現代の計算機ではすべての候補を検討することも不可能ではないですが、10^{24} ともなると効率的に検討しないと到底最善にたどり着けません。

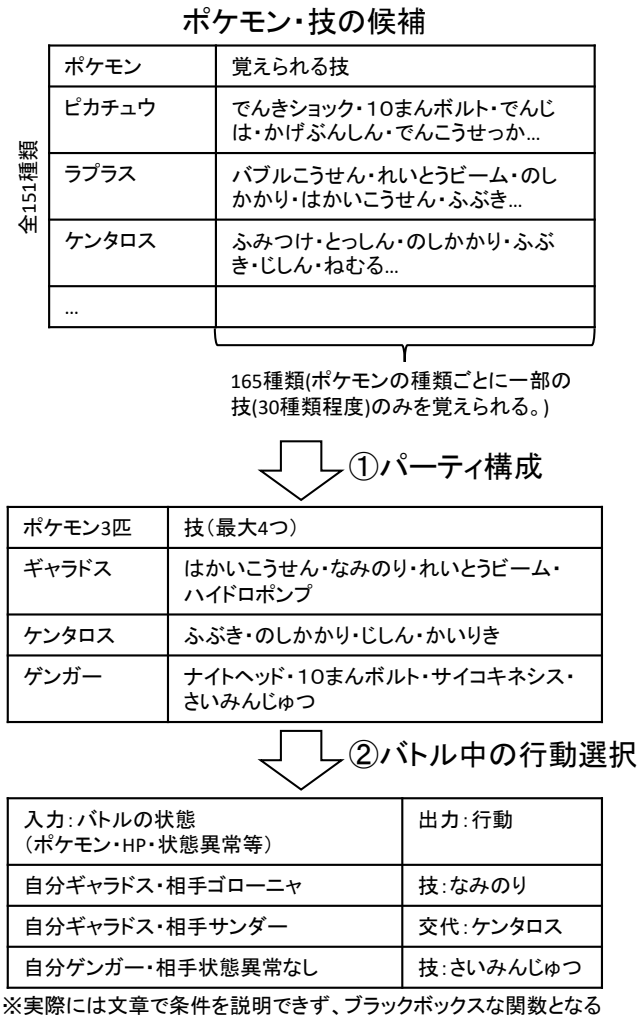
次にバトル中の行動選択です。1vs1 であれば、1 回の行動は 4 種類の技のどれかを選ぶものでした。3vs3 では、場に出ているポケモンの覚えている技 4 種類のどれかを選ぶか、控えのポケモン最大 2 匹のどれかに交換するという 6 種類の選択肢が生じます。技を選ぶ場合ですが、1 番目のポケモンの 1 番目の技と、2 番目のポケモンの 1 番目の技は全く関係がありません。単純に選択肢が 1.5 倍になるより複雑で、場に出ているポケモンに応じて選択肢の選び方を変える能力が必要であるといえます。また、自分のポケモンが瀕

[*1] 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。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をパーティに入れるかどうかの選択もキーとなります。

[*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。

[*3] 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`

死になった際は、次のターンに進む前に控えのポケモンのうちどれを出すかを選ぶ必要があります。この状況への対処も新たな課題となります。



▲図 1.1　ポケモンバトル AI の 2 つの課題。

1.2 ポケモンバトルのシステム

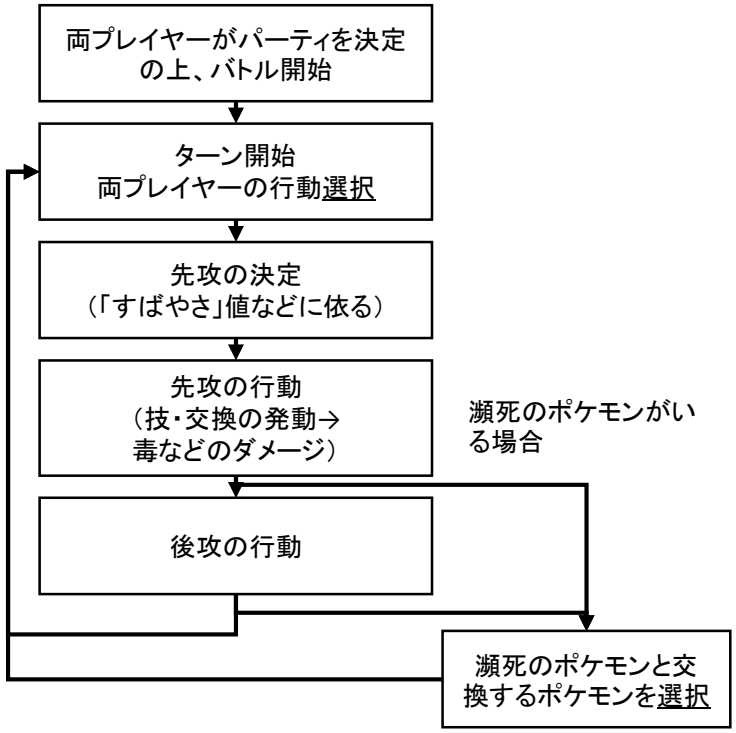
第 1 巻の再掲となりますが、ポケモンバトルの基本的なシステムについておさらいしておきましょう。ゲームのバージョンにより少し異なりますが、初代バージョンに準拠します。また、ここで説明しきれないさまざまな例外的状況がありますが、省略しています。

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、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。「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。

本書では、ポケモンバトルのルールを独自に Python 言語で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。実装上の制限から、一部原作ソフトと挙動が異なる点があります。状況が複雑化するため、次の技は未実装です：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。また、技の使用回数制限（PP）はありません。ミュウツー・ミュウはパーティに含めません。

第 2 章

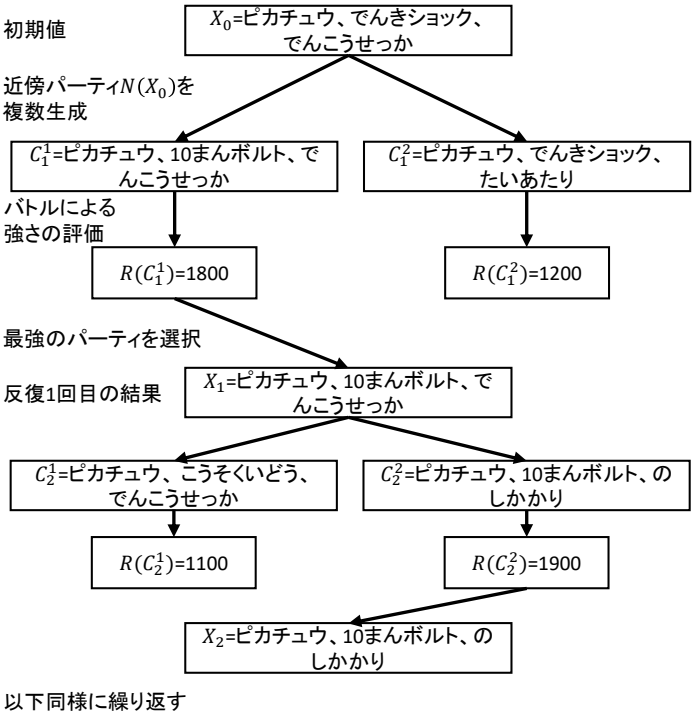
パーティ構成の最適化手法

本書ではパーティ構成とバトル中の行動選択の両方の自動化を目的としています。第 1 巻では 1vs1 バトルにおいてパーティ構成と行動選択を交互に最適化する手法を提案しました。しかしながら 3vs3 ではそれぞれの計算コストが増大し、同じ手法をとることが困難でした。そのため、まず強いパーティを生成し、その行動方策を学習するという順序で目的を実現します。

2.1　第 1 巻のおさらいと 3vs3 への拡張

第 1 巻では、ランダムなパーティから開始して、近傍パーティを複数生成、それらの強さを測定し一番強いものを取り出すという処理を反復的に行う山登り法を提案しました。概要を図 2.1 に示します。

まず、初期値としてランダムなパーティ X_0 を生成します。パーティの「近傍」(Neighbor) をランダム生成する関数 N を定義し、ステップ t におけるパーティ X_t を入力として近傍パーティ $C_{t+1}^1, C_{t+1}^2, ..., C_{t+1}^K$ を生成します。パーティの「強さ」(Rate) を測定する関数 R を定義し、先ほど生成されたパーティのうち最強のもの $X_{t+1} = \mathrm{argmax}_i R(C_{t+1}^i)$ を選択します。この処理を X_{t+1} に対しても反復的に行うことで、徐々に強いパーティが生成されていきます。繰り返し回数は T で表すことにします。一括でランダムなパーティを KT 個生成するよりも、ステップを分けて徐々に強くしていくほうが、同じ候補数でもより強いパーティが得られることがわかっています。ここで必要となる関数が 2 つあり、設計が必要です。1 つはパーティの近傍を生成する関数 N で、もう一つが強さを測定する関数 R です。



▲図 2.1　山登り法の概要

2.1.1　パーティの近傍生成

パーティの近傍を生成する関数 N は、入力されたパーティがある程度有望だとして、それをあまり変えないでもう少し強いパーティの候補を生成するためのものです。パーティの要素はポケモンと技なので、これを少し変更するということになります。1vs1 バトルの場合は、ポケモンの技を 1 つ選択し、ポケモンが覚えられる技の範囲で別の技に変更するという操作を用いました。パーティのポケモンが 1 匹だけなので、ポケモンを変更する操作はできません。なぜならポケモンによって覚えられる技が異なるため、ポケモンを変更すると現在の技を残すことができず、パーティの全要素を変更することになり近傍ではなくなってしまうためです。一方今回の 3vs3 バトルではポケモンが 3 匹いるので、そのうち 1 匹を別のポケモン・技に置き換えたとしても近傍として変更前のパーティの性質を保つことができます。そのため操作は 2 種類としました。1 つはポケモンを 1 匹選び、別のポケモンに置き換えます。新しいポケモンの技は、ランダムに決定します。も

う 1 つの操作はポケモンを 1 匹選び、その中の技を 1 つ選び、別の技に置き換えます。技は、当然対象ポケモンが覚えられる範囲で選択します。なお、ルールとしてポケモン 1 匹のレベルは 50〜55 の間で変動することになっていますが、この配分は初期値生成の時に固定し、その後変化させないこととしました。レベル 50 を 2 匹とレベル 55 を 1 匹生成し、その順序はランダムとします。

2.1.2 強さの測定

強さを測定する関数 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 とします。

レートの計算をするためには勝率が必要で、バトル 1 回だけで計算はできません。実験

では、パーティのレートの推定値を更新しながら同程度の強さのパーティと対戦を繰り返し、100 回のバトルでレートの計算結果を出力します。1 回のバトルは数 ms かかるため、1 回のレート計算に 0.1〜1 秒程度の時間がかかることになります。これを山登り法の繰り返し処理に組み込んで数 100 回行うと数分かかります。さらに、ポケモンには相性があるため、絶対的に強い単一のパーティがあるわけではありません。条件 A ではたまたま水タイプポケモンばかりのパーティ、条件 B では電気タイプポケモンばかりのパーティができたとすると、条件 B は条件 A より良いということになりますが、パーティ 1 つだけで判断するのは危険だといえます。そのため同じ実験条件で乱数を変更して 100 個のパーティを生成し、異なる条件のパーティを混合してレート計算をしたうえで、ある条件で生成されたパーティの平均レートをもってその条件（パーティ生成・行動選択のアルゴリズムやハイパーパラメータ）の良さを測定します。これらのことから、現実的な計算時間の範囲では山登り法での繰り返しの回数があまり増やせません。3vs3 では、1vs1 と比べてパーティの空間が広いため、予備実験において明らかに最適化不十分となってしまいました。また、1vs1 の時は、山登り法の途中で生成されたパーティそれぞれに対して、バトルにおけるパーティの行動をランダムではなく強化学習によって最適化したうえでレート計算をするという手法を提案し、良い結果を得ました。しかし 3vs3 では強化学習自体の計算時間を長くとる必要があり、さらに山登り法の繰り返し回数も増やす必要があるとなると、一般的なパソコンで計算できる範囲を超えてしまいました。

2.2 パーティ評価関数の学習

山登り法で有望なパーティを計算する際のバトルに時間がかかるなら、バトルせずに強さを予測できればよさそうです。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、もっと高速な手法でこの結果 R_b 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 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。今回は、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 151 種類、技は 165 種類あるため、2 要素の組み合わせとなると組み合わせが比較的多く、合計 50086 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

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

特徴	なみのり	じばく	じしん	かたくなる
値	1	0	1	0

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

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

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 50086 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。

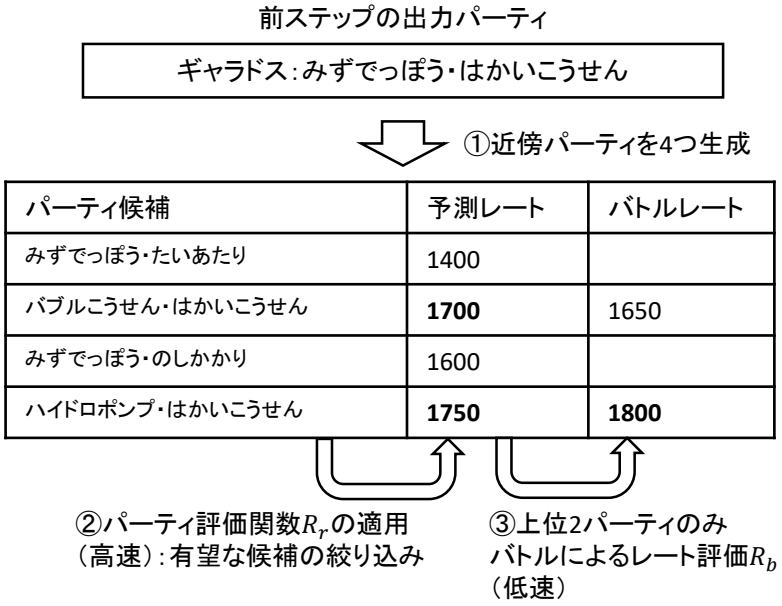
バトル不要で強さを予測するという目標なのに、学習データとしてパーティをバトルさせてレート計算させるというのは奇妙に聞こえるかもしれません。しかし、パーティ 10000 個のレートをバトルで計算してモデルを学習してしまえば、そのモデルを用いてあらゆる未知のパーティの強さを予測できます。もちろん誤差は出るのですが、潜在的には 10^{24} 個のパーティが存在するにもかかわらず、そのほんのわずかだけを学習対象にすればよいため、十分に効率的だといえます。これを汎化能力といい、これを実現するのが機械学習の大きな目標です。

2.3 パーティ評価関数を用いた強いパーティの生成

前節で述べたように、パーティに対してその強さを予測する関数ができたとして、それを用いて最強のパーティを生成する手法を検討します。

1 つ目の手法は、既存の山登り法を少し変更します。既存の方法は、現在のパーティの近傍をランダムに生成し、それらをすべてバトルによって評価して最強のパーティを選択していました。ここにパーティ評価関数を使い、有望なパーティを絞り込むステップを導入します。例えば、既存の山登り法ではランダムな近傍パーティを 10 個生成してバトルで評価していたとします。これに対し、ランダムな近傍パーティを 100 個生成し、それらをパーティ評価関数を用いて強さを予測、上位 10 個だけをバトルで評価します。パーティ候補の生成数が増えるため、より強力なパーティが得られる確率が高まると期待できます。バトルが数 ms かかるところ、予測関数は μs のオーダーで計算できるため、この変更による計算時間の増加はわずかです。これは、最良優先探索という枠組みに含まれます。図 2.3 に最良優先探索の 1 ステップの簡単な例を示します。前のステップの出力パーティに対し、近傍パーティ 4 つを生成します。次にパーティ評価関数により、これらのレートを予測します。その結果の上位 2 パーティにのみ、時間がかかるバトルでのレート計測を行い、最強のパーティを出力します。パーティ候補として上の 2 つだけを生成してバトルで評価する場合よりも、ほぼ同じ計算コストでよい結果を得ることに成功しています。あくまでパーティ評価関数は近似でしかないという考え方で、予測されたレートとバ

トルでのレートの大小関係が逆転する可能性もあるため、バトルによる評価を行っています。ただし、あまりにも予測が外れている場合には本来強い候補を除外してしまう可能性もあり、欠点となります。



▲図 2.3　最良優先探索 1 ステップの例。簡略化のためパーティのポケモンは 1 匹・技は 2 つの場合で表示。

2 つ目の手法は、バトルを一切行わず、パーティ評価関数が最大となるパーティ $\mathrm{argmax}_X R_r(X)$ を求めるものです。もしパーティ評価関数の近似性能が十分高いのであれば、パーティ評価関数が最大値を取るパーティがバトルにおいても最強であると考えられます。そのようなパーティを求める具体的なアルゴリズムとして、同様に山登り法が使えます。強さをバトルで評価していたのを、パーティ評価関数で評価するように変更すれば実装できます。ただし、山登り法は真の最適解を求めてくれる保証がないことに注意してください。

第 3 章

パーティ構成の実験

前章で提案した手法の効果を実験で確認します。まずパーティ評価関数を学習させ、それに基づいてパーティを生成します。

3.1　パーティ評価関数の学習

提案のパーティ生成手法で必要となる、対戦なしにパーティの強さ（レート）を推定するパーティ評価関数の学習とその結果の評価を行います。

評価関数の学習データとしてパーティを 10000 個ランダムに生成し、それらを相互に対戦させてレートを算出しました。パーティとレートの例をリスト 3.1 に示します。上位パーティにゲンガーが多く登場することや、下位パーティにだいばくはつを覚えたポケモンがいることが見て取れます。このような傾向を学習し、パーティの構成要素の善し悪しを判断できるようにすることが目標となります。ただ、学習データ内で上位 2 番目のパーティでも、そんなに強くないと考えられるパラスを含んでいることに注意が必要です。ランダムに生成したパーティなので、強力なポケモンだけですべて固めるということはできていません。これは学習データに含まれるノイズの一種だと考えられるでしょう。

▼リスト 3.1　パーティ評価関数の学習データの例。最上位・最下位 3 パーティを表示。レートの平均は 1500。レート 2000 はレート 1500 のパーティに 95% 勝利する強さ。

```
rate=2042
ゲンガー（LV 50）
  ちきゅうなげ サイコキネシス かみなり 10 まんボルト
ジュゴン（LV 55）
  みずでっぽう ふぶき ねむる はかいこうせん
クラブ（LV 50）
  かげぶんしん すてみタックル どくどく かたくなる

rate=2019
```

```
オニドリル（LV 50）
  みだれづき ゴッドバード はかいこうせん スピードスター
ゲンガー（LV 55）
  あやしいひかり じごくぐるま 10まんボルト ナイトヘッド
パラス（LV 50）
  かげぶんしん つるぎのまい メガドレイン ひっかく

rate=2005
ゲンガー（LV 55）
  かみなり はかいこうせん サイコウェーブ ちきゅうなげ
キングラー（LV 50）
  どくどく すてみタックル のしかかり バブルこうせん
フリーザー（LV 50）
  ねむる かまいたち バブルこうせん すてみタックル

...

rate=875
アーボ（LV 55）
  へびにらみ にらみつける どくどく かげぶんしん
ゴース（LV 50）
  だいばくはつ したでなめる かげぶんしん どくどく
コクーン（LV 50）
  かたくなる いとをはく どくばり

rate=863
コイキング（LV 55）
  たいあたり はねる
ヒトカゲ（LV 50）
  リフレクター なきごえ りゅうのいかり ロケットずつき
キャタピー（LV 50）
  たいあたり いとをはく

rate=855
タマタマ（LV 50）
  だいばくはつ かげぶんしん しびれごな すてみタックル
イシツブテ（LV 50）
  メガトンパンチ ねむる まるくなる だいばくはつ
コクーン（LV 55）
  いとをはく かたくなる どくばり
```

これらのパーティとレートを学習データとし、パーティ評価関数を学習しました。ライブラリとして scikit-learn の `sklearn.svm.LinearSVR`を用いました。正則化パラメータ`C`については、5-fold cross validation により $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ から最適なものを選択しました。比較として、特徴量の一部のみを使用した場合の回帰精度を計算しました。

学習結果のモデルの回帰精度を表 3.1 に示します。すべての特徴量を使用することで最も高い予測精度が実現されました。

学習されたパラメータの定性的解釈を行いました。線形モデルなので、各特徴量に対するパラメータがレートへの寄与だと考えることができます。

パラメータごとの係数を、特徴量の種類ごとに分けて表 3.2 に示します。正の大きな係

▼表 3.1 特徴量に対するパーティ評価関数の回帰精度。R^2 が高いほど良い。P: ポケモン、M: 技 (move) を表し、PP 等はその組み合わせを表す。

P	M	PP	MM	PM	R^2 値
○					0.514
	○				0.480
○	○				0.746
		○			0.209
○		○			0.486
			○		0.428
	○		○		0.490
○	○	○	○		0.732
○	○	○	○	○	**0.764**

数はレートを上げるのに寄与している要素、負の大きな係数はレートを下げるのに寄与している要素であるとみなせます。

P 特徴量を見てみます。ゲンガー・ケンタロスなど初代を代表する強力なポケモンが上位に来ており、妥当な結果です。最下位はプリンでした。直観的にはコイキングやトランセルが最下位になるはずです。しかしこれらのポケモンは、はねるやかたくなるといった特徴的な技を必ず持っているため、M 特徴量でそれらの技に低い係数が割り当てられることにより弱さが表現されているようです。普通の技を覚えるポケモンの中で、比較的弱いポケモンが P 特徴量の最下位に来ているという結果だと考えられます。

M 特徴量を見てみます。サイコキネシス・ふぶきといった汎用的で強力な技が最上位です。はなびらのまいやあばれるなど、少し意外な技もあります。これらは一度発動すると複数ターン連続で自動的に攻撃し続けるという特徴があります。技の選択がランダムなので、変な技を選ぶよりは一度選ぶとある程度強力な攻撃が継続する技がより高評価なのかもしれません。最下位はじばくでした。ランダムに使えば自滅を招く技なので妥当なところでしょう。

PP 特徴量を見てみます。全体的に係数が小さく、レート推定にあまり寄与していないようです。交換を絡めた戦術を使えない以上、相性補完などが評価されていないのでしょう。学習データが 10000 パーティ程度なので、1 つの特徴が使われるのは平均 3 回とかなり少ないのも要因でしょう。

MM 特徴量を見てみます。だいばくはつ, じばくの組が最もレートを上げるという結果になっています。これはどういうことでしょうか？ M 特徴量においてだいばくはつもじばくも非常に低い係数になっています。しかしこれらを両方覚えていても 1 つ覚えているのとほとんど変わらないといえます。そのため、下がりすぎたレートを補正する役割を

担っているとみなせます。このように特徴量の間に相関関係があるため、必ずしも見た目通り解釈できない部分があります。そのほかでは、攻撃技と補助技の組み合わせや、タイプの異なる攻撃技の組み合わせが高い係数のようです。

PM 特徴量を見てみます。スターミーとサイコキネシスの組のように、タイプ一致技が高い係数になっています。ポケモン・技単体の効果では表しきれない、タイプ一致によるダメージ増加の効果を表現できているものと考えられます。逆に最下位では、強力なポケモンがだいばくはつを覚えている場合や、とくしゅが低いポケモンが特殊技を覚えている場合などが表れているようです。

以上のような形で、パーティを与えるとレートを推定してくれるモデルを学習することができました。なお、自爆技が最低の係数になっていることからも明らかなように、「ランダムに行動する」という前提条件の下で学習されたモデルであることに注意する必要があります。

▲表 3.2　各種特徴量に対するレート回帰係数。最大・最小 10 件。

P特徴	係数
ゲンガー	0.379
ケンタロス	0.375
サンダー	0.338
ギャラドス	0.311
ラプラス	0.297
スターミー	0.273
カイリュー	0.269
フリーザー	0.263
カビゴン	0.247
ガルーラ	0.238
…	
マンキー	−0.221
ニドラン♂	−0.224
カラカラ	−0.224
コラッタ	−0.226
ゼニガメ	−0.239
アーボ	−0.242
ミニリュウ	−0.248
コダック	−0.250
ニョロモ	−0.261
プリン	−0.278

M特徴	係数
サイコキネシス	0.272
ふぶき	0.250
れいとうビーム	0.235
はなびらのまい	0.234
ハイドロポンプ	0.222
はかいこうせん	0.207
あばれる	0.201
なみのり	0.197
ナイトヘッド	0.195
かえんほうしゃ	0.192
…	
ひかりのかべ	−0.159
どくばり	−0.162
しっぽをふる	−0.167
なきごえ	−0.181
こうそくいどう	−0.203
はねる	−0.207
テレポート	−0.208
だいばくはつ	−0.278
いとをはく	−0.295
じばく	−0.326

PP特徴	係数
ニドクイン,ナゾノクサ	0.079
ケーシィ,ギャラドス	0.060
ガーディ,プテラ	0.059
ヒトカゲ,カブトプス	0.057
ギャロップ,ゴースト	0.053
ウツボット,サンダース	0.052
モンジャラ,プテラ	0.051
ベロリンガ,ケンタロス	0.051
ギャロップ,レアコイル	0.051
コダック,カイロス	0.050
…	
ヤドン,ビリリダマ	−0.052
ズバット,パラス	−0.054
フシギダネ,メノクラゲ	−0.054
ニャース,ヒトデマン	−0.054
イワーク,ドガース	−0.054
ラフレシア,アズマオウ	−0.054
コクーン,ベロリンガ	−0.055
プリン,コダック	−0.060
オニスズメ,ビリリダマ	−0.060
パラセクト,コイル	−0.074

MM特徴	係数
だいばくはつ,じばく	0.166
ナイトヘッド,どくどく	0.132
はかいこうせん,なみのり	0.126
ふぶき,スピードスター	0.126
さいみんじゅつ,10まんボルト	0.122
のしかかり,つるぎのまい	0.115
さいみんじゅつ,サイコキネシス	0.110
ゆめくい,メガドレイン	0.110
すいとる,ねむりごな	0.107
ふぶき,のしかかり	0.106
…	
だいばくはつ,サイコウェーブ	−0.131
なきごえ,しっぽをふる	−0.143
リフレクター,しっぽをふる	−0.144
どくばり,いとをはく	−0.156
ナイトヘッド,じばく	−0.156
メガドレイン,じばく	−0.158
ねむる,じばく	−0.163
はねる,たいあたり	−0.198
いとをはく,たいあたり	−0.216
だいばくはつ,ねむる	−0.220

PM特徴	係数
スターミー,サイコキネシス	0.226
フリーザー,ふぶき	0.197
フリーザー,れいとうビーム	0.192
サンダー,10まんボルト	0.188
シャワーズ,なみのり	0.173
フーディン,サイケこうせん	0.172
ラプラス,れいとうビーム	0.169
フーディン,サイコキネシス	0.168
オムスター,なみのり	0.165
サンダース,かみなり	0.164
…	
コンパン,サイコキネシス	−0.147
ゲンガー,だいばくはつ	−0.148
ビードル,どくばり	−0.150
ビードル,いとをはく	−0.150
ゴース,じばく	−0.153
コラッタ,バブルこうせん	−0.165
マンキー,10まんボルト	−0.166
コイキング,たいあたり	−0.198
コイキング,はねる	−0.198
ゲンガー,じばく	−0.241

3.2 パーティの生成

前節で学習されたパーティ評価関数を用いて、パーティの生成を行います。手法として、ランダムに生成したもの、山登り法により最適化したもの、パーティ評価関数に基づき最適化したものを比較します。様々な条件で生成したパーティ（各条件 100 パーティ）を 1 つの環境で相互に対戦させ、平均レートを計算しました。

本書を通じた強さの基準として、ランダムに生成した 10000 パーティを相互に対戦させ、レートを事前に付与しました。バトル中の行動はランダムです。環境に他の様々な条件で生成したパーティを投入しレートを計算しますが、このパーティのレートは変動させず、常に平均 1500 を保つものとします。

各種条件を示します。

- (A) 山登り法 (10,10): 1 iteration（反復）あたり 10 パーティ、10 iteration。パーティ評価関数を用いない手法です。
- (B) 最良優先 (10,100,10): 1 iteration あたり近傍 100 個生成、評価関数で上位 10 パーティ選択、それをバトルでレーティング、10 iteration。(A) とバトルの回数は同じで、有望な候補に絞ったバトルを行います。
- (C) 評価関数のみ (100,10): 1 iteration あたり近傍 100 個生成、評価関数で上位 1 パーティ選択、10 iteration。(B) と生成される候補数は同じですが、ベストなパーティを決める際にバトルを用いません。
- (D) 評価関数のみ (100,1000): 1 iteration あたり近傍 100 個生成、評価関数で上位 1 パーティ選択、1000 iteration。とにかく評価関数が最大値を取るようなパーティを生成します。
- (E) 評価関数のみ (10,10): 1 iteration あたり近傍 10 個生成、評価関数で上位 1 パーティ選択、10 iteration。(A) と候補数は同じで、バトルの代わりに評価関数を用います。

全候補数の比較では、(A) 山登り法 (10,10)=(E) 評価関数のみ (10,10)<(B) 最良優先 (10,100,10)=(C) 評価関数のみ (100,10)<<(D) 評価関数のみ (100,1000) となります。バトル回数をコストとすると、(A) 山登り法 (10,10)=(B) 最良優先 (10,100,10) となります。他はバトルを行いません。

結果を表 3.3 に示します。同じ候補数の (A) と (E)、(B) と (C) を比較すると、どちらも評価関数で強さを測定するほうがバトルで測定するより強いパーティが生成できるという結果になりました。バトルでの測定は本来の目的に一番合致しているとはいえ、技の命中をはじめとしたランダム性に左右されます。そのため、安定性が高い評価関数のほうが

▼表 3.3　パーティ生成条件と、平均レート

条件	総候補パーティ数	バトルによる測定回数	平均レート
(R) ランダム	-	-	1500
(A) 山登り法 (10,10)	100	100	1743
(B) 最良優先 (10,100,10)	1000	100	1942
(C) 評価関数のみ (100,10)	1000	0	2011
(D) 評価関数のみ (100,1000)	100000	0	**2101**
(E) 評価関数のみ (10,10)	100	0	1856

よかったのかもしれません。一番強かったのは、評価関数をとにかく最大化する条件 (D) でした。今回の実験条件においては、評価関数は探索の補助役ではなくそれ自体を最適化の目標にできるレベルの精度があったといえます。

条件 (D) で生成された最強パーティをリスト 3.2 に列挙します。

▼リスト 3.2　条件 (D) におけるの最強パーティ TOP5。

```
rate=2207
ギャラドス (LV 50)
  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 (LV 50)
  ふぶき のしかかり じしん かいりき
ゲンガー (LV 55)
  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

rate=2199
サンダー (LV 50)
  そらをとぶ ドリルくちばし かみなり はかいこうせん
フリーザー (LV 55)
  ねむる れいとうビーム ゴッドバード ふぶき
ゲンガー (LV 50)
  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

rate=2192
サンダー (LV 50)
  かみなり 10まんボルト そらをとぶ ドリルくちばし
ギャラドス (LV 50)
  ハイドロポンプ れいとうビーム なみのり はかいこうせん
ゲンガー (LV 55)
  サイコキネシス さいみんじゅつ ちきゅうなげ ナイトヘッド

rate=2191
ゲンガー (LV 50)
  メガドレイン 10まんボルト さいみんじゅつ サイコキネシス
オムスター (LV 50)
  はかいこうせん れいとうビーム ハイドロポンプ なみのり
ゴースト (LV 55)
  ナイトヘッド サイコウェーブ あやしいひかり かみなり
```

```
rate=2188
ラプラス（LV 55）
  なみのり はかいこうせん ハイドロポンプ れいとうビーム
フリーザー（LV 50）
  ふぶき ねむる れいとうビーム ゴッドバード
ゲンガー（LV 50）
  10まんボルト さいみんじゅつ ナイトヘッド サイコキネシス
```

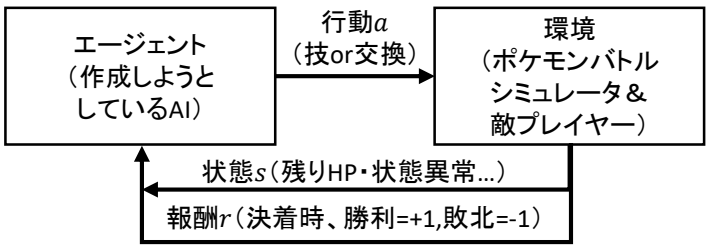
比較的強力そうなパーティができたといえます。強力なポケモンおよびそのポケモンにあった技を覚えていることが確認できます。しかし、ラプラスが同系統の「なみのり」「ハイドロポンプ」を両方覚えている点は改善の余地がありそうです。評価関数の学習法を改善すれば、MM 特徴として同系統の技同士が共存することに対して負のスコアを与えることで回避できるかもしれません。一方で手法の本質的な課題として、評価関数を最大化するパーティは 1 つかごく少数しかなく、最適化がうまくいけばいくほど同じパーティに収束してしまうという課題があります。ポケモンの種類はなんとかばらけているものの、ゲンガーやフリーザーの技は異なるパーティでも全く同じになってしまっています。パーティの多様性確保は今後の課題と言えます。また、ランダムな行動をした場合の強さを元に評価関数を学習しているため、使いどころが難しい補助技は少なめです。

第 4 章

バトル中の行動選択モジュールの強化学習

4.1　強化学習の枠組みとポケモンバトルへの応用

生成されたパーティに対して、バトル中の行動選択モジュールを学習します。このモジュールは、バトルの各ターンにおいて、ポケモンの残り HP や状態異常などの状態を受け取り、次の行動すなわち繰り出す技や交換を決定します。どんな状態においてどの行動を選択するか、という判断基準を複数ターンからなるバトルの最後に決まる勝敗から学習する必要があります。このような課題を解決する一般的な枠組みとして「強化学習」があります。



▲図 4.1　強化学習の一般的な枠組みと、ポケモンバトルにおける意味（カッコ内）。

その枠組みとポケモンバトルにあてはめた場合の意味を図 4.1 に示します。エージェントが今回作ろうとしているモジュールに相当します。ターンの開始時に、環境がバトルの状態 s を送ってきます。それをもとに、エージェントは行動 a を選択し、環境に送りま

す。環境は敵プレイヤーの行動を選択し、ポケモンバトルシミュレータを用いて実際に行動（技や交換）を発動させ、バトルを 1 ターン進めます。そして新たな状態 s' をエージェントに送ります。同時に、報酬 r もエージェントに送ります。報酬とは、そのターンで新たに得られたゲームの得点です。ポケモンの場合は、ゲームの決着がついた（どちらかのプレイヤーの持つポケモンが全滅した）ときに、勝利なら +1 点、敗北なら-1 点が得られます。決着がつかなかったターンでは 0 点です。強化学習の目的は、ゲームが終了するまでに得られる得点を最大化することです。強化学習が教師あり学習と異なっているのは、行動 1 つ 1 つに正解・不正解が与えられておらず、1 ターンごとの報酬ではなく、ゲーム全体の合計報酬を最大化しなければならないという点です。ポケモンバトルで言えば、さいみんじゅつなどの補助技を使った場合、そのターンに勝利するということはありませんので、直後の報酬は 0 です。その後に攻撃技を使って勝利して初めて、報酬が得られます。補助技が役立つからといって、さいみんじゅつだけを連打していては勝てません。補助技を使った後の行動に依存し、遅れてやってくる報酬を用いて、補助技に効果があったかどうかを判断しなければならないという点が難しい点となります。

囲碁やインベーダーゲームなど多くのゲームでは、1 種類のゲームに対して 1 つの最強エージェントを学習させることが一般的です。これらのゲームでは、ゲームの初期状態は基本的に一定です。しかしポケモンバトルでは、自分のパーティ構成に幅広い選択の余地があるという特徴があります。本研究では、パーティごとに固有の行動選択パラメータを学習する方針をとります。あらゆる手持ちパーティに対して適用できるパラメータを学習しようとすると、100 種類以上のポケモンや技すべての運用法を表現できる規模の大きなモデルが必要となり、学習の所要時間や安定性が問題になることが予測されるためです。

強化学習の 1 つの定式化である行動価値関数ベースの手法では、「ある状態 s においてある行動 a を取ったときに、将来的にもらえる報酬の和の期待値」を表す Q 関数 $Q(s,a)$ を定義し、その関数のパラメータを最適化します。Q 関数を使った行動の選択は、$a = \mathrm{argmax}_a Q(s,a)$、すなわち現在の状態において Q 関数の値が最大となる行動 a を選択するということです。Q 関数の形式は Deep Neural Network を用いたモデルとします。このモデルは、状態と行動の関係性を複雑な非線形関数であらわすことができます。Q 関数の入力は s, a として表現されていますが、実際には $Q(s)$ という形式の関数が行動の数に対応した（18 次元の）ベクトルを出力し、各次元の値が a に対応するという実装になります。これは Deep Neural Network で Q 関数を学習する DQN[*1]という枠組みで一般的に用いられる手法です。

[*1] V. Mnih et al., Playing Atari with Deep Reinforcement Learning. In NIPS Deep Learning Workshop 2013.

自分：ゲンガー（10まんボルト・さいみんじゅつ）
相手：ギャラドス or ケンタロス（どちらもじしんのみ）

10まんボルト：ギャラドスを1発、ケンタロスを2発で倒せる。
さいみんじゅつ：60%で相手を眠らせる。起きることはない。
ゲンガーが常に先制する。
相手は常にじしんを使用し、ゲンガーを1発で倒せる。

環境の条件

対ギャラドス：10まんボルトを使用。
対ケンタロス：最初にさいみんじゅつを使用し、次に10まんボルトを2回使用。さいみんじゅつが外れれば敗北。命中率60%なので、さいみんじゅつを使用するタイミングでのQ値は
1(勝利)×60%+(-1)(敗北)×40%=0.2となる。

戦略

s_1：相手が水タイプ（ギャラドス）かどうか
s_2：相手が状態異常（ねむり）どうか

入力特徴の定義（2次元）

技	Q関数
10まんボルト	$\theta_{11}s_1 + \theta_{12}s_2 + \theta_{13}$
さいみんじゅつ	$\theta_{21}s_1 + \theta_{22}s_2 + \theta_{23}$

Q関数（線形モデル）の定義
パラメータθは6要素を持つ

$\theta_{11} = 2$	$\theta_{12} = 2$	$\theta_{13} = -1$
$\theta_{21} = 0$	$\theta_{22} = 0$	$\theta_{23} = 0.2$

パラメータの例（強化学習手法により最適化する対象）

相手	状態異常	Q値計算結果 （上：10まんボルト、 下：さいみんじゅつ）
ギャラドス	なし	$2\times1+2\times0-1=\mathbf{1}$ $0\times1+0\times0+0.2=0.2$
ギャラドス	ねむり	$2\times1+2\times1-1=\mathbf{3}$ $0\times1+0\times1+0.2=0.2$
ケンタロス	なし	$2\times0+2\times0-1=-1$ $0\times0+0\times0+0.2=\mathbf{0.2}$
ケンタロス	ねむり	$2\times0+2\times1-1=\mathbf{1}$ $0\times0+0\times1+0.2=0.2$

表現力の限界により1を超えてしまっている

ここで10まんボルトを使うと敗北

Q値最大となる技がエージェントの行動として出力される

▲図 4.2　Q 関数を用いたエージェントの単純化された例。

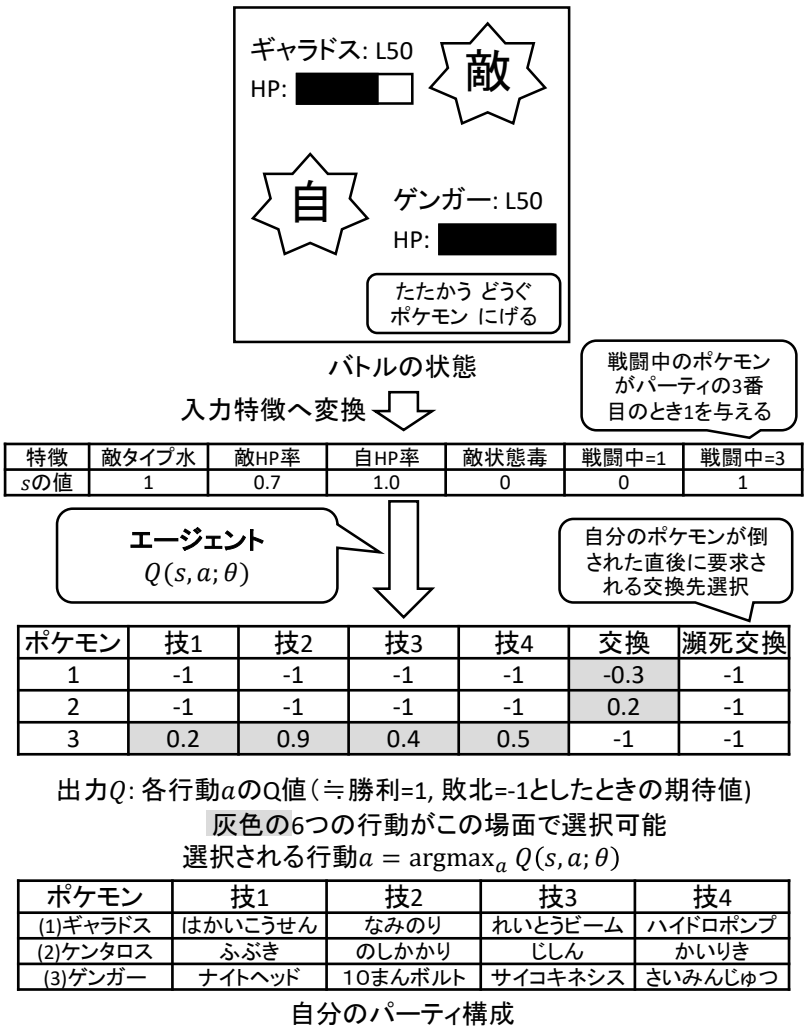
説明が抽象的なため、図 4.2 にて非常に単純化した例で Q 関数を用いたエージェントの働きを説明します。ここでは相手が 2 種類しか存在しないと仮定し、それぞれを倒すのに異なる戦略が必要となっています。入力特徴から相手が何であるか、どういう状況にあるかを判断して、Q 値を出力します。ここで登場するパラメータ θ は、この例では筆者が手作業で埋めていますが、これを報酬すなわちバトルの勝敗を用いて決定することが強化学習の目的となります。

4.2 入出力の設計

1vs1 バトルだと、行動 a の空間は技 4 種類のどれか 1 つを選ぶというものでした。3vs3 では、技 4 種類のほか、控えのポケモンに交換する行動が追加されます。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換は 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、3vs3 バトルの場合はターン開始時の行動選択として技か交換を選ぶだけでなく、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換があります。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、そのポケモンに交換する場合、瀕死時にそのポケモンに交換する場合の 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 4.3 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 Q 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。Q 関数の出力として、各行動 a ごとの Q 値が得られます。そして、Q 値が最大となる行動を選択し、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 種類か交換・瀕死時の交換の 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。もしこれに対応する Q 値が全行動中最大となり、その行動が選ばれてしまうと問題になります。エージェントがれいとうビームを選んだつもりでも、シミュレータはそれに従って動かない（別の何らかの有効な行動を選択したことにして進めることにするほかない）ため認識にずれが生じてしまうためです。

入力特徴については、場に出ている自分・相手ポケモンの状態とともに、自分のポケモ



▲図 4.3　エージェントの入出力。

ンの 3 匹中どれが場に出ているか、控えを含めてどのポケモンが瀕死でないかを表す次元を追加しました。この情報を正しく Q 関数が考慮できれば、現在の状態でどの行動が可能なのかということがわかります。表 4.1 に入力特徴の全体図を示します。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

▼表 4.1 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
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（なし・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・すばやさ・とくしゅ・命中・回避それぞれ、ランク/12+0.5 を設定）
場インデックス	3	自分	パーティ 3 匹のうち現在場に出ているポケモンに該当する次元に 1 を設定
生存インデックス	3	自分	パーティ 3 匹のうち瀕死でないポケモンに該当する次元に 1 を設定
ポケモンタイプ	15	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

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（151 次元）ではなく、タイプ（15 次元）にしています。自分側のポケモンのタイプは入れていません。あくまで特定のパーティを操作するためだけのパラメータを学習しているため、場に出ているポケモンのインデックスがあれば、自分側のポケモンは一意に特定でき、相性の情報はわかるはずだからです。1 点注意したいのは、ポケモンのタイプの意味について何ら AI は事前知識を持っていません。人間であれば水タイプの技は炎タイプのポケモンによく効くという設定はしっくりきますが、AI にとってはそのようなイメージを想起させる作用はまったくなく、タイプ 1、タイプ 2…というようなただの数値にすぎません。技についても、なみのりが水タイプということは知らず、あくまで技 1、技 2…の 4 種類があるだけです。技の効果や相性の情報はすべてバトルの勝敗から学習していく必要があります。

4.3 報酬の設計

ここまでは Q 関数をブラックボックスとして考えた場合の入出力の話でした。強化学習アルゴリズムにより Q 関数のパラメータを学習させようと思った場合、1vs1 より難しくなります。単純にバトルのターン数が増加するため、バトルの勝敗に対し、どの行動がよかったのか悪かったのかを判断することが難しくなります。また、1 番目のポケモンが場に出ているときに 3 番目のポケモンの技を選ぶような出力をした場合、それは無効なので選ばないように誘導する必要があります。これら 2 点を解決するため、強化学習時の報酬に工夫を加えました。

1 点目の課題については、あるターンで与えたダメージを直後の報酬として与えるようにしました。それにより、各ターンの行動でできるだけ多くのダメージを与えるよう誘導できます。単純なダメージだと数値が大きい（勝敗を +1,-1 としているため）うえ、ポケモンの種類により幅が大きく異なるため、最大 HP との比率で計算します。さらに、相手ポケモンを倒した場合は確実に勝利に近づいているため、単純なダメージだけでなく倒したことそのものにも報酬を与えるべきと考えます。相手からダメージを受けないようにするための補助技にもインセンティブを与えたいので、相手からダメージを受けた場合は負の報酬を与えるようにしました。全体として、H=(相手の残り HP/相手の最大 HP+ 相手の残りポケモン数)-(自分の残り HP/自分の最大 HP+ 自分の残りポケモン数) とおいて、「ターン開始時の H - ターン終了時の H」× 0.5 を報酬にしました。それでも、補助技や交換を行ったターンはこちらから与えるダメージがないため、報酬が小さいという問題は残ります。

2 点目の課題については、選択できない行動を選択した場合には負の報酬-0.1 を与えることにしました。これにより有効な行動の中で強いものを選んでくれることを期待します。

4.4 対戦相手の設計

一人プレイのゲーム、例えばインベーダーゲームであれば、強化学習の環境は一定です。しかし囲碁やポケモンなど、対戦ゲームにおいては環境の一部として対戦相手が必要であり、相手の強さによって適切な戦略が異なるという課題が存在します。相手が非常に弱く、例えばトランセルでかたくなるを連打してくるような状況だと、どんな行動をしようが勝ててしまいます。このような環境で強化学習を行っても戦略間に差が出ず、賢くない戦略が出てくる可能性が高いといえます。そのため対戦相手も強力なパーティ・強化学習によるエージェントを用いる必要があると考えられます。今回使用した汎用的な強化学

習フレームワーク ChainerRL では、この点をサポートする機能は存在しないようです。そのため本プロジェクト用の機能として対戦相手として学習済みの強化学習エージェントを使うための実装を行いました。

第 5 章

バトル中の行動選択モジュールの実験

提案手法を用いてエージェントを強化学習しました。実験条件として、3 つの条件を変えて評価しています。1 つ目の条件は、エージェントが用いるパーティです。ランダム生成されたパーティと、3 章でパーティ評価関数を用いて生成された強力なパーティを比較します。2 つ目の条件は、学習時の報酬です。勝敗だけでなくダメージ・選択できない行動に関する報酬を加えた場合とそうでない場合を比較します。3 つ目の条件は、学習時の対戦相手です。パーティの種類および、行動がランダムか強化学習されたものかを変化させ、比較します。最初はランダムに行動するエージェントしかありませんので、それを対戦相手に強化学習したエージェントを学習します。次にそのエージェントを対戦相手に別のエージェントを学習します。1 つのエージェントだけを相手にしてしまうとその相手を倒す戦略しか学習できないので、対戦 1 回ごとに相手となるエージェントをランダムに選択します（後述のように、1 つの生成条件で 100 個の異なるエージェントが学習されるので、それらをローテーションさせる）。

エージェントの Q 関数のモデルは Deep Neural Network で、バトルの状態を表すベクトルを入力とし、18 種類の行動それぞれに対する Q 値を出力します。ネットワーク構造は全結合層のみ、隠れ層 2 層、各 32 チャンネルとしました。モデル構造の変更を試みましたが、試した範囲では、モデルを小さくすると弱くなり、大きくしても変化がありませんでした。学習手法は DQN の改良版である DoubleDQN[*1]、探索手法は ConstantEpsilonGreedy（学習中、現在のモデルが選択した最適な行動を 80% の確率で用い、20% の確率でそれを無視してランダムに行動します）、最適化手法は Adam としました。学習中、モデルが選択した行動が無効である場合があります。この場合でも何らか

[*1] H. Hasselt et al., Deep Reinforcement Learning with Double Q-learning. In AAAI 2016.

の行動をとってターンを進めないといけないので、有効な行動のうち先頭のものが選ばれたものとみなして進行させるようにしました。ターン開始時の行動選択では先頭の技、瀕死時の交換ではポケモンリストのうち瀕死でない最初のポケモンを選択することに対応します。これをランダムに選択するように変更しても強さは向上しませんでした。

DoubleDQN を実装した深層強化学習フレームワークとして ChainerRL を用いました。GPU は使わず、すべて CPU 上で計算を行っています。

各種条件において 100 個ずつエージェントを学習しました。同じ条件の中で、エージェントが用いるパーティがすべて異なっています。そして、すべてのエージェントおよびベンチマーク用のエージェントを相互に対戦させ、エージェントごとのレートを計算します。

▼表 5.1 エージェントの学習条件と、平均レート。パーティは、ランダム (Random) は「r」、評価関数 (Evaluation function) を用いて生成されたものは「e」。敵エージェントは、ランダムパーティかつランダム行動は「rr」、評価関数によるパーティかつランダム行動は「er」、強化学習で学習されたものは「et」(条件 Pe-Eer-Rdi のエージェントを使用)。条件 ID は、パーティの条件「P」、敵 (Enemy) の条件「E」、報酬 (Reward) の条件「R」の組み合わせで決定。報酬の「d」は、ターンごとに与えたダメージにより報酬を与える条件、「i」は無効 (illegal) な行動により負の報酬を与える条件。

条件 ID	パーティ	敵エージェント	ダメージ報酬	無効行動報酬	平均レート
(ベンチマーク用)	ランダム	-	-	-	1500
Pr-Err-Rdi	r	rr	○	○	1768
Pr-Err-Ri	r	rr		○	1761
Pr-Err-Rd	r	rr	○		1756
Pr-Eer-Rdi	r	er	○	○	1661
Pr-Eer-Ri	r	er		○	1556
Pr-Eer-Rd	r	er	○		1616
Pe-Err-Rdi	e	rr	○	○	2044
Pe-Err-Ri	e	rr		○	2034
Pe-Err-Rd	e	rr	○		2032
Pe-Eer-Rdi	e	er	○	○	2073
Pe-Eer-Ri	e	er		○	2078
Pe-Eer-Rd	e	er	○		2068
Pr-Eet-Rdi	r	Pe-Eer-Rdi	○	○	1573
Pr-Eet-Ri	r	Pe-Eer-Rdi		○	1451
Pr-Eet-Rd	r	Pe-Eer-Rdi	○		1507
Pe-Eet-Rdi	e	Pe-Eer-Rdi	○	○	**2107**
Pe-Eet-Ri	e	Pe-Eer-Rdi		○	2060
Pe-Eet-Rd	e	Pe-Eer-Rdi	○		2082

条件ごとにエージェントのレートを平均して表 5.1 に表示します。組み合わせが複雑なため、略号を用いています。最も強かったのは、条件 Pe-Eet-Rdi、すなわち強力なパーティを用いて、強力な敵と対戦しながら学習、学習時の報酬としてダメージ・無効な行動に関するものを加えたものが最も強いという結果でした。個別の条件を見ていきます。P{r,e}-Err-Rdi を比較すると、パーティの生成方法の違いが強さに最も影響していることがわかります。ポケモンや技が弱いと立ち回りではどうしようもないというのは当然の結果と言えます。Pr-E{rr,er,et}-Rdi 間を比較すると、自分のパーティや報酬の条件が同じでも、相手が強いほど強力な戦略を学習できていることがわかります。下手な行動が即負けにつながるからでしょう。Pe-Eet-R{di,i,d}を比較すると、報酬は提案した 2 種類両方を用いることで若干強さが向上することがわかります。特に、Pr-Eet-R{di,i}間の差は顕著です。これらの条件では、エージェントはランダムなパーティを用いているのに対し、相手は強力なパーティ・戦略を用いてくるため、ほとんどの場合負けてしまうと考えられます。このような条件ではどんな行動をとろうが負けるため、エージェントが「すねて」しまう可能性が高いです。しかし、ダメージに対して報酬が与えられることで、負けるという結果の中でも極力多くのダメージを与えるように学習できていると考えられます。Pr-E{rr,et}-Rdi の比較でもわかるように、自分のパーティや報酬の条件が同じでも、敵が強すぎるとかえってエージェントが弱くなってしまうという現象がみられます。対戦相手の設定は、エージェントが勝てる見込みがある範囲内でできるだけ強くすることが重要だといえます。

最も強力なエージェントが生成できる条件 Pe-Eet-Rdi において生成されたエージェントのうち、最もレートが高かったエージェントが使っていたパーティをリスト 5.1 に示します。

▼リスト 5.1　強化学習結果の最強エージェント（Pe-Eet-Rdi）が用いていたパーティ TOP3。

```
rate=2297
ゲンガー（LV 55）
ナイトヘッド さいみんじゅつ 10 まんボルト サイコキネシス
スターミー（LV 50）
じこさいせい なみのり はかいこうせん ハイドロポンプ
フリーザー（LV 50）
れいとうビーム ねむる ふぶき ゴッドバード

rate=2274
ギャラドス（LV 50）
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（LV 50）
ふぶき のしかかり じしん かいりき
ゲンガー（LV 55）
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

rate=2271
```

```
カビゴン（LV 50）
のしかかり じしん なみのり はかいこうせん
サンダー（LV 55）
かみなり そらをとぶ 10まんボルト ドリルくちばし
ドククラゲ（LV 50）
ハイドロポンプ バブルこうせん れいとうビーム ふぶき
```

順当に強そうなパーティが並んでいます。実際のバトルのログを 1 つ図 5.1 に示します。パーティ 1 のゲンガーは相手のタイプや状態異常を考慮して適切な技を選ぶことに成功しています。実際にはここまでうまくいく例は少なく、半減される技を選んだり、すでに眠っている相手にさらにさいみんじゅつを使おうとしたりする場合もあります。また、相手にポケモンを倒されたときに負の報酬が発生するため、HP が減って倒される直前のポケモンを交換するという戦略も時折見られました。ほとんどの場合は被害を拡大するだけの結果になってしまうため、適切でない戦略となります。ダメージや倒したことに対する報酬を与える手法の欠点と言えます。安定して戦略を学習できる技術が確立したとはまだいえない状況です。

最も悪い条件 Pr-Eet-Ri での学習結果を見ていると、変わった戦法がありました。パーティをリスト 5.2 に示します。このパーティはパラスが場に出た時、キノコのほうしを使い続けます。キノコのほうしは 100% 命中して相手を眠らせる技です。しかも初代のシステムでは、ポケモンが目覚めたターンには行動できないため、目覚めた直後にキノコのほうしを受けるとなにもできません。この技を覚えるのはパラス・パラセクトだけで、素早さが低く打たれ弱いため、通常は相手に先手を取られて攻撃されて一発で瀕死となります。しかし相手の攻撃が外れたり補助技が使われたりすれば、この無限ループに持ち込むことができます。今回実装したシステムでは、PP が無限かつ 64 ターンで引き分けという原作にない条件が設定されており、エージェントはこの穴を突くことで「負けない」戦略を実現したようです。

▼リスト 5.2　学習状態の悪いエージェント（Pr-Eet-Ri）が用いていたパーティの一例。

```
rate=1515
ニドキング（LV 50）
じわれ はかいこうせん じごくぐるま いわなだれ
ニドラン♂（LV 55）
かげぶんしん どくばり たいあたり とっしん
パラス（LV 50）
ねむる せいちょう キノコのほうし ソーラービーム
```

残念ながら、ポケモンの交換を積極的に活用した戦略というのは見られませんでした。相手を眠らせていない限り、交換の隙にダメージを受けるため短期的には損な戦略になるためでしょう。また、初代のルールでは場に対する効果を及ぼす技がない（リフレクター

パーティ1				
ポケモン	技0	技1	技2	技3
ゲンガー	ナイトヘッド	さいみんじゅつ	10まんボルト	サイコキネシス
スターミー	じこさいせい	なみのり	はかいこうせん	ハイドロポンプ
フリーザー	れいとうビーム	ねむる	ふぶき	ゴッドバード
パーティ2				
ポケモン	技0	技1	技2	技3
ラプラス	なみのり	はかいこうせん	ハイドロポンプ	れいとうビーム
フリーザー	ふぶき	ねむる	れいとうビーム	ゴッドバード
ゲンガー	10まんボルト	さいみんじゅつ	ナイトヘッド	サイコキネシス

パーティ1		パーティ2
ターン1		
ゲンガーを繰り出した		ラプラスを繰り出した
ゲンガーの10まんボルト	→	ラプラス HP259→64
ゲンガー HP167→106	←	ラプラスのハイドロポンプ
ターン2		
ゲンガーの10まんボルト	→	ラプラス HP64→0
		フリーザーを繰り出した
ターン3		
ゲンガーのさいみんじゅつ	→	フリーザーは眠った
		眠っている
ターン4		
		ゲンガーを繰り出した
ゲンガーの10まんボルト	→	ゲンガー HP167→117
ターン5		
ゲンガーのサイコキネシス	→	ゲンガー HP117→29
ゲンガー HP106→36	←	ゲンガーのサイコキネシス
ターン6		
ゲンガーのサイコキネシス	→	ゲンガー HP29→0
		フリーザーを繰り出した
ターン7		
ゲンガーの10まんボルト	→	フリーザー HP197→98
		眠っている
ターン8		
ゲンガーの10まんボルト	→	フリーザー HP98→0
バトル終了 勝者 パーティ1		

眠らせる→効果抜群の技というコンボを想定。相手が交換したので決まらず。

相手のタイプが変わったので技を変更。

素早さが同じなので、乱数によってはパーティ1側が倒れていた。

▲図 5.1　強化学習エージェント同士のバトルの一例。

等も交換すると効果が消えます）ため、場を整えるための補助技だけを持ったポケモンというのは存在できず、有力な戦略として浮上することは難しいと思われます。

5.1　97 年大会で用いられた補助技

実験結果として、補助技のあるパーティを生成し、それを正確に運用するのはまだ難しいという状況でした。

補助技は多数あるものの、そのすべてがバトルで有用とはいえません。どのような補助技に活用の余地があるのか知るため、初代ポケモンを使った大会において活用された技を調べました。ニンテンドウカップ'97 の全国大会出場者 15 名のパーティ（合計 90 匹）[*2]に含まれる攻撃技・補助技を列挙しました。

攻撃技は以下の 31 種類でした。１０まんボルト・あなをほる・いわなだれ・かいりき・かみなり・きりさく・サイケこうせん・サイコキネシス・じごくぐるま・じしん・そらをとぶ・だいばくはつ・だいもんじ・タマゴばくだん・ちきゅうなげ・でんこうせっか・ドリルくちばし・ナイトヘッド・なみのり・にどげり・のしかかり・ハイドロポンプ・はかいこうせん・はっぱカッター・バブルこうせん・ふぶき・ミサイルばり・メガドレイン・ゆめくい・れいとうパンチ・れいとうビーム。

補助技は以下の 18 種類でした。あくまのキッス・あやしいひかり・かげぶんしん・くろいきり・こうそくいどう・さいみんじゅつ・じこさいせい・たまごうみ・ちいさくなる・テレポート・でんじは・どくどく・ねむりごな・ねむる・ひかりのかべ・へんしん・やどりぎのタネ・リフレクター。

大会中にどの程度活用されたかまではわかりませんが、補助技がある程度採用されていたことは間違いないようです。さすがにテレポートは全くの無意味ですが。状態異常系のほか、回避率上昇、回復技が使われています。当時は攻撃力・特殊を上げる技が乏しく、かげぶんしん以外の積み技は用いられていないようです。AI でも、少なくとも状態異常系と回避率上昇は使いこなせるよう改良の余地があると考えられます。

[*2] http://www2u.biglobe.ne.jp/~kakeru/pokemon/ps/97.htm

第 6 章

まとめ

本書では、初代ルール、3vs3 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。パーティの生成については、ポケモンや技がパーティの強さに貢献するか否かを得点付けするパーティ評価関数を学習し、その関数を元に強力なパーティを生成しました。生成されたパーティを用いたバトル中の行動選択については、強化学習を用い、ポケモンの交換を考慮すること、学習時の報酬としてダメージを考慮すること、対戦相手として強いエージェントを用いることで、強力なエージェントを生成しました。

パーティの生成では、各ポケモンにあった適切な攻撃技が選択されました。一方、ランダムに技を繰り出す前提でのパーティ構成となったため、補助技は比較的選ばれにくい結果となりました。バトル中の行動選択における強化学習からのフィードバックを受けられる機構が今後必要と言えます。バトル中の行動選択では、ある程度パーティの強さを引き出せていました。一方、タイプ相性でいまひとつな技を選択したり、すでに眠っている相手へさいみんじゅつを選択してしまうなど、荒のある結果となりました。学習の余地がかなり残されているといえます。

今後の展望としては、バトルの展開の先読みを行うことが挙げられるでしょう。ある技を出した後にどんな状態になるか、その状態がどの程度好ましい（勝利に近い）かということを数ターン後まで予測することです。囲碁のようにプレイヤーが交互に行動し、ランダム性のないゲームでは十分研究されていますが、ポケモンのように同時に行動し、ランダム性や相手しか知らない情報（まだ繰り出していないポケモンや技）があるゲームに対してどう応用するかはかなり難しい問題です。また、イロレーティングによる強さ測定の枠組みも完ぺきとは言えません。パーティ間に相性があるため、一直線上にパーティを並べるだけではパーティ間の強さ比較ができない側面があります。様々なタイプや戦略を持つパーティを明示的に生成できるような手法が必要と考えます。この課題について、パー

ティの強さをスカラー値ではなくベクトル値で表現する手法[*1]が提案されており、ポケモンバトルへの応用を考慮すべきでしょう。

学習環境として、初代ポケモンはかなりゲームバランスに難があるというのも事実です。ふぶきが強すぎたり、エスパータイプに弱点が実質ないなどの問題や、複数のポケモンでの連携をとる選択肢に乏しいなどの要因があります。よほど AI が賢くない限り、単純で十分強い戦略があると、そこで学習が止まってしまいます。手の込んだ戦略にスポットライトを当てるための手段として、ポケモン金銀のルールを採用することも考慮したいところです。

[*1] D. Balduzzi et al., Re-evaluating Evaluation. In NeurIPS 2018.

付録 A

付録

A.1　ふぶきの凍らせる確率とパーティの強さに与える影響

初代ポケモンにおいて、技「ふぶき」は極めて強力な技として知られています。ポケモン 151 種類の「初代」の括りのなかでも、ソフトによってその強さに変更が加えられています。最初に発売されたソフトであるゲームボーイの赤・緑では、ふぶきの追加効果で相手を凍らせる確率は 30% でした。凍らされたポケモンは一切行動できず、相手があえて（炎タイプの技を使うなどの）ミスをしない限り溶けることもないため実質的な瀕死状態といえます。ポケモンスタジアム[*1]以降では、ふぶきで凍らせる確率は 10% に修正されています。さらに第 2 世代（金・銀）以降は、凍っているポケモンは 1 ターンごとに一定の確率で溶けて行動できるようになりました。

それでは、PokéAI の世界でふぶきの設定を変化させた場合、どの程度影響があるのでしょうか。凍らせる確率を 30% と 10% で変化させて実験してみました。パーティ（1 パーティは 3 匹のポケモンを含む）をランダムに 10000 種類生成しました。そのうち 1881 パーティがふぶきを持っていました。2 つの条件でパーティ同士を戦わせ、レーティングを計算しました。行動の選択はランダムです。ふぶきは強力ですが、高々 1/4 の確率でしか使われません。別々の環境での対戦なのでレート同士を直接比較ないため、各環境の中で、各パーティのレートを偏差値として算出、ふぶきを持っているパーティの偏差値平均を計算しました。

結果を表 A.1 に示します。若干の差ではありますが、ふぶきの強力さと、ふぶきを持つパーティの強さに正の相関があるという結果になりました。ふぶきが強力すぎると戦略が画一化される恐れがあるため、本書の実験はふぶきにより凍らせる確率を 10% に設定し

[*1] ポケモンスタジアムは据え置きゲーム機であるニンテンドウ６４で、３Ｄグラフィックでポケモンバトルを楽しめるゲームソフト。ゲームボーイソフトである赤・緑を接続し、セーブデータ中のポケモンを使って対戦することも可能。テレビにポケモンバトルを映せるため、大会でも用いられた。

▼表 A.1　ふぶきにより凍らせる確率と、ふぶきを持つパーティのレート偏差値平均

確率	偏差値の平均
30%	54.8
10%	53.6

て実験を行うこととしました。

A.2　初代で「さかさバトル」

PokéAI の目的として、人間が具体的な戦略を与えず、コンピュータの計算によって戦略を学習するというものがあります。この条件は人間の先入観によって戦略を狭めないというだけでなく、未知のルールでの最適な戦略を発見することに役立ちます。本書では、初代のルールで「さかさバトル」をしたらどんなパーティや戦略が強いのかということを実験しました。

さかさバトルとは、第 6 世代（X・Y）で導入された特殊なバトルルールで、タイプ相性が反転するというものです。通常のルールでは炎タイプの技は水タイプのポケモンに対してこうかはいまひとつ（ダメージ 1/2 倍）ですが、さかさバトルではダメージ 2 倍となります。地面タイプの技を飛行タイプのポケモンに対して打った場合のように、通常 0 倍となる場合もさかさバトルでは 2 倍となります。通常ダメージとさかさバトルでのダメージについて全パターンを示すと、2 倍⇒ 1/2 倍、1 倍⇒ 1 倍、1/2 倍⇒ 2 倍、0 倍⇒ 2 倍となります。タイプ一致ボーナスは変わりません。

このルールを初代のルールに導入して実験してみます。筆者の予想としては、ノーマルタイプのケンタロスがノーマルタイプの技「はかいこうせん」を連打する戦略が最強と予想しました。与えるダメージを半減されたり無効化されたりすることがなく、威力 150 の技を出せるのは非常に強力だと考えられます。

本編で実証したパイプラインを、ルール変更した状態で動作させました。シミュレータの実装にリスト A.1 のように変更を加えることで実装しました。

▼リスト A.1　さかさバトル実現用コード

```
# 通常の相性設定。ここでは、炎タイプの技について各タイプのポケモンへのダメージ倍率を設定。
# 1=1/2 倍、4=2 倍。
_type_match_dict[PokeType.FIRE] = {
    PokeType.FIRE: 1,
    PokeType.WATER: 1,
    PokeType.GRASS: 4,
    PokeType.ICE: 4,
    PokeType.BUG: 4,
```

```
    PokeType.ROCK: 1,
    PokeType.DRAGON: 1,
}

# さかさバトルの場合の相性書き換え。
def _sakasa_type_match():
    for d in _type_match_dict.values():
        for k in d.keys():
            d[k] = [4, 4, 2, 2, 1][d[k]]

# 環境変数が設定されていた時に相性の書き換えを実行。
if os.environ.get("POKEAI_SAKASA", None) == "1":
    _sakasa_type_match()
```

まずはパーティ評価関数の学習を行いました。パラメータごとの係数を、特徴量の種類ごとに分けて表 A.2 に示します。正の大きな係数はレートを上げるのに寄与している要素、負の大きな係数はレートを下げるのに寄与している要素であるとみなせます。

P 特徴を見ると、ノーマルタイプのケンタロス・ガルーラがトップです。氷・エスパー弱点がなくなったことで、ニドキング・ニドクインが上位に来ています。「にらみつける」でおなじみのファイヤーが 10 位です。伝説ポケモンとして能力値は高いので、タイプ相性さえなんとかなれば活躍できるのかもしれません。下位については通常ルールとほとんど変わりません。能力値の低さは相性とは関係がないためです。M 特徴を見ると、かえんほうしゃ・はなびらのまいがトップです。通常ルールではまず使われないほのおタイプ・くさタイプが生きてくるようです。PP・MM 特徴はやはりよくわからない結果です。PM 特徴ではタイプ一致技が上位です。これは通常ルールと変わらない結果です。

パーティ評価関数を用いてパーティを生成、そして強化学習を行いました。その結果の最強パーティをリスト A.2 に示します。

▼リスト A.2　さかさバトルで、強化学習結果の最強エージェントが用いていたパーティ

```
rate=2289
フリーザー (LV 55)
ふぶき バブルこうせん れいとうビーム ねむる
ケンタロス (LV 50)
だいもんじ はかいこうせん ふみつけ のしかかり
ガルーラ (LV 50)
じしん すてみタックル メガトンキック ちきゅうなげ

rate=2276
フリーザー (LV 55)
ゴッドバード バブルこうせん れいとうビーム ねむる
ケンタロス (LV 50)
だいもんじ ふぶき ねむる かいりき
ペルシアン (LV 50)
はかいこうせん 10 まんボルト きりさく のしかかり

rate=2264
```

```
フリーザー（LV 50）
れいとうビーム ねむる バブルこうせん ふぶき
スターミー（LV 50）
なみのり サイコキネシス ハイドロポンプ じこさいせい
ケンタロス（LV 55）
すてみタックル だいもんじ のしかかり はかいこうせん
```

通常ルールと変わらず、フリーザーは強力です。相手を凍らせることの強さは変わらないからでしょう。ケンタロスがだいもんじを覚えているところが特徴的です。水タイプ対策でしょうか。トップ 2 つのパーティはどちらも、ノーマルタイプのポケモン 2 匹をパーティに入れています。初代さかさバトルではノーマルタイプが最強というのが今回の結論です。AI のアルゴリズムが改良されれば、それに従って最適解も変わっていくと予想されます。

▲表 A.2　さかさバトルにおける、各種特徴量に対するレート回帰係数。最大・最小 10 件。

P特徴	係数
ケンタロス	0.430
ガルーラ	0.345
ギャラドス	0.320
フリーザー	0.319
ラプラス	0.294
ニドキング	0.290
ニドクイン	0.287
カビゴン	0.287
カイリュー	0.287
ファイヤー	0.271
…	
ニドラン♂	−0.221
ケーシィ	−0.221
ヤドン	−0.225
アーボ	−0.225
ヒトカゲ	−0.230
コダック	−0.232
ゼニガメ	−0.235
プリン	−0.239
カブト	−0.249
ミニリュウ	−0.265

M特徴	係数
かえんほうしゃ	0.294
はなびらのまい	0.254
はかいこうせん	0.245
ハイドロポンプ	0.236
だいもんじ	0.232
あばれる	0.230
なみのり	0.226
ふぶき	0.193
きりさく	0.163
はっぱカッター	0.163
…	
なきごえ	−0.169
ほえる	−0.170
こうそくいどう	−0.174
しっぽをふる	−0.178
ふきとばし	−0.185
はねる	−0.188
ひかりのかべ	−0.200
じばく	−0.233
テレポート	−0.244
いとをはく	−0.313

PP特徴	係数
フシギバナ,ガーディ	0.057
オニスズメ,スターミー	0.056
ピジョット,カビゴン	0.054
ズバット,カイロス	0.053
ワンリキー,サンダース	0.050
レアコイル,サイホーン	0.050
マダツボミ,カイリュー	0.050
フシギソウ,サンダー	0.050
ユンゲラー,サイドン	0.050
ニドクイン,ミニリュウ	0.050
…	
ヤドン,レアコイル	−0.053
ビードル,ズバット	−0.055
コクーン,ベロリンガ	−0.055
ポッポ,パラス	−0.056
サンドパン,ニョロモ	−0.058
ラプラス,ファイヤー	−0.058
ビードル,アーボ	−0.059
バタフリー,イシツブテ	−0.059
オニスズメ,クラブ	−0.059
ディグダ,コイル	−0.063

MM特徴	係数
ひのこ,だいもんじ	0.139
のしかかり,はかいこうせん	0.133
かげぶんしん,だいもんじ	0.131
だいばくはつ,じばく	0.122
のしかかり,つるぎのまい	0.108
サイコキネシス,ちきゅうなげ	0.107
れいとうビーム,リフレクター	0.104
はなびらのまい,しびれごな	0.103
すてみタックル,かえんほうしゃ	0.101
はなびらのまい,つるぎのまい	0.101
…	
なきごえ,つるぎのまい	−0.118
こうそくいどう,リフレクター	−0.131
かげぶんしん,なきごえ	−0.132
こうそくいどう,ふきとばし	−0.140
せいちょう,つるぎのまい	−0.144
どくばり,いとをはく	−0.148
だいばくはつ,ねむる	−0.152
ねむる,じばく	−0.159
はねる,たいあたり	−0.186
いとをはく,たいあたり	−0.215

PM特徴	係数
フリーザー,れいとうビーム	0.248
サンダー,10まんボルト	0.213
フーディン,サイコキネシス	0.193
フリーザー,ふぶき	0.185
レアコイル,10まんボルト	0.182
ガルーラ,メガトンキック	0.182
ドードリオ,のしかかり	0.177
リザードン,だいもんじ	0.169
フリーザー,ねむる	0.169
シャワーズ,なみのり	0.166
…	
イシツブテ,だいもんじ	−0.133
ケーシィ,ねむる	−0.135
クラブ,ねむる	−0.139
ビードル,どくばり	−0.142
ビードル,いとをはく	−0.142
ズバット,すてみタックル	−0.142
キャタピー,たいあたり	−0.157
キャタピー,いとをはく	−0.157
コイキング,たいあたり	−0.186
コイキング,はねる	−0.186

あとがき

昨年 10 月から半年かけて、第 2 巻をお届けできました。アイデア自体は昔から考えていたので、そこでの苦労は少なかったのですが、いざ実装して動作させようとすると試行錯誤が必要で時間がかかってしまいました。本文中で述べたように問題点がいくつか判明したので、対策を立てていきたいと考えています。

今後の方向性ですが、大きく分けて 2 つあります。1 つはアルゴリズムの改良で、もう 1 つはポケモン金銀環境への移行です。補助技を絡めた戦術こそ知的な感じがするので、それをより実現しやすい環境を設けてみたいと思います。

ところで、ポケモンの新作「ソード・シールド」が発表されましたね。筆者もぜひプレイしたいです。最近はプライベートの時間もゲーム AI の研究に多くの時間を割いていて、ゲーム自体を人力でプレイすることがかなり減ってしまっています。人工知能の有名な研究者 Andrej Karpathy（の同僚）の名言があります[*2]。

> Coworker on RL research: "We were supposed to make AI do all the work and we play games but we do all the work and the AI is playing games!"

ソード・シールドが発売されたときには研究の手を止めて、久しぶりにゲームに没頭したいですね。

本書の校正には noe さんに協力していただきました。ありがとうございました。

第 1 巻の刊行時、タイトルを Pok「è」 AI としていたのですが、壮大な誤植でした。「ポケモン」に対してはアクセントの向きが違う「é」が正しいです。訂正させていただきます。

本書の電子版を `https://select766.booth.pm/items/1294595` 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 `27wqfRWJGq` です。

[*2] `https://twitter.com/karpathy/status/784212363062816768`

ＰｏｋéＡＩ　＃２：初代３ｖｓ３編

2019 年 4 月 14 日　技術書典６ v1.2.0

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