強化学習を利用したAIディレクターの制作

濱石海地

2021年7月20日

1　はじめに

AIディレクターとはゲームのおいて進行管理役を務めるシステムであり、ゲームを遊ぶプレイヤーの状態を監視し、ゲームの難易度をリアルタイムで調整を行う。

ゲームにおいて難易度は、それ一つでゲームの面白さを左右する重要な要素であり、ゲーム開発において最も工数をかけるべき箇所のひとつである。しかしユーザーにとって面白く感じる難易度は人それぞれであり、また工数の多さは開発者にとって大きな負担となる。当研究は「AIディレクター」に着目し、これを機械学習を用いて作成することでユーザーそれぞれに最適な難易度を提供すると共に、開発者の負担を軽減できる可能性を狙ったものである。

2　研究の概要

自作のゲームにおいて、ゲームに登場する敵やアイテムの出現を強化学習AIに制御させる。

3　ゲームの概要

Python製の、コマンドライン上で動作する簡素なRPG。

3.0.1　ゲーム全体の流れ

以下の流れを繰り返すことで進行する。これを所定の回数繰り返した後、プレイヤーが生存しているならゲームクリアとする。

1. 2通りの行先が提示されるので、プレイヤーはそれを選ぶ。行先に何があるかは表示されている。強化学習AIはここで何が提示されるのかを選ぶ。

2. 選んだ行先に敵がいるなら、それと戦う。アイテムがあるなら、それを獲得する。

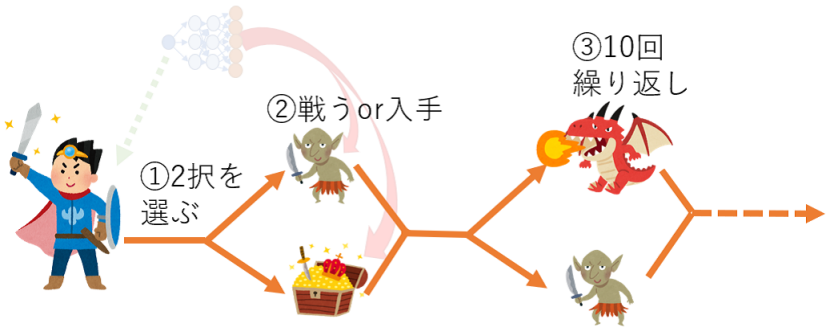


図 1: ゲームを遊ぶ流れのイメージ図

3.0.2 戦闘の流れ

戦闘ではプレイヤーと敵が、どちらかの生命力が 0 以下になるまで、「プレイヤー、敵、プレイヤー、敵…」の順番で交互に行動する。戦闘における行動は以下の通り。

表 1: 戦闘における行動

攻撃	相手に自身の攻撃力と同値のダメージを与える (生命力を減らす)。
防御	相手から受けるダメージを、自身の防御力の値だけダメージを減少させる。
薬草を使用	薬草所持数を 1 消費し、自身の生命力を全回復。
爆弾を使用	爆弾所持数を 1 消費し、攻撃力 2 倍の攻撃を行う。

3.0.3 プレイヤーおよび敵のステータス (能力値)

プレイヤーおよび敵には、以下のステータスが存在する。

3.1 強化学習 AI の概要

Chainer を利用したニューラルネットワークを使用し、Deep Q-Network と呼ばれる方法をとる。入力層には、プレイヤーのステータス (整数値)6 個と、選択肢を選び進んだ回数 (0 20 の整数値) の、計 7 個の値が入力される。出力

表 2: プレイヤーおよび敵のステータス (能力値)

生命力	これが 0 になったら死亡。アイテム等により回復する。
最大生命力	生命力はこの値を超えて回復しない。アイテム拾得により増加。
攻撃力	戦闘において敵に与えるダメージ量に影響。 戦闘に勝利するか、特定のアイテム拾得により増加。
防御力	戦闘において防御したとき、この値だけ受けるダメージ量が減る。 戦闘に勝利するか、特定のアイテム拾得により増加。
薬草所持数	戦闘中に消費して生命力を回復できる。アイテム拾得により増加。
爆弾所持数	戦闘中に消費して攻撃力 2 倍の攻撃ができる。アイテム拾得により増加。

層には、出現する全ての敵 11 種類と、出現する全アイテム 6 種類、それぞれの出現確率を決める実数値 17 個が出力される。

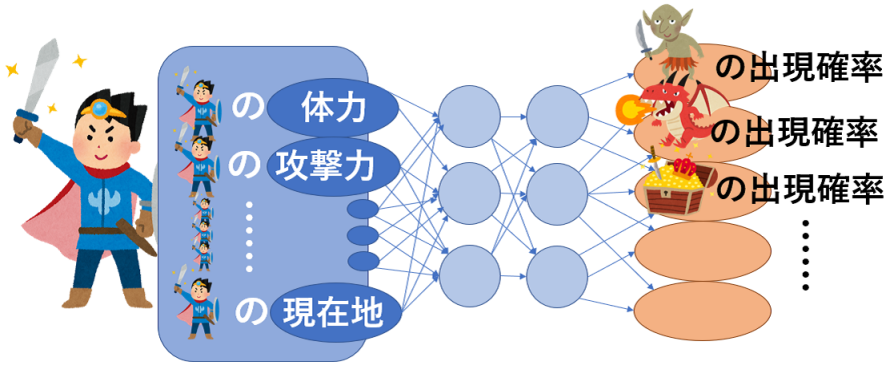


図 2: ディレクターの仕組み

学習するにあたって、ルールベースで自動的にゲームを遊ぶプレイヤーを使用する。当研究において、学習するのはプレイヤー (遊ぶ側) ではなく、ディレクター (遊ばれる側) であることに留意。

3.1.1 学習データ

学習に使用する学習データは、遊んだ時の状況を再現したものを入力値に、ゲームを 10 回プレイした際の成績から算出したスコアを出力値とする。これを教師あり学習と同じように学習する。

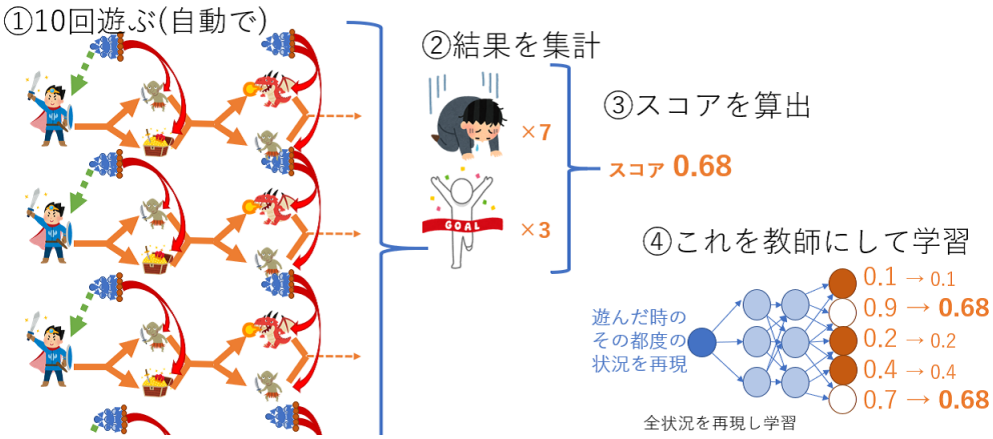


図 3: 学習全体の流れ。図中の数値は研究初期の一例

4　学習の結果と考察

4.1　一般的な ε -グリーディ法を使用した方法

まず最初に一般的な強化学習の手法に見られるような、出力値のうち最も高い値 2 つを選択肢として提示する方法をとった。特徴は次の通りである。

4.1.1　スコアの算出方法

プレイヤーが死亡するまでに到達したゲームの進行度 10 回分の平均と、定めた目標の差をスコアとしている。現在は 14 回目の選択肢でちょうど力尽きる難易度に調整する (先に記した通り、20 回目の選択肢を生き抜いたらゲームクリア) ために、目標の値を 14 としている。このときスコアは-14 0 の範囲となり、0 が最高である。

4.1.2　Fixed Target Q-Network

ネットワークを毎回更新すると、いつまで経っても収束しないという事態が発生する可能性があるため、学習はスコアの取得 10 回毎にのみ行う。それまで教師データは保存される。

4.1.3　ε に基づくランダムなアクション

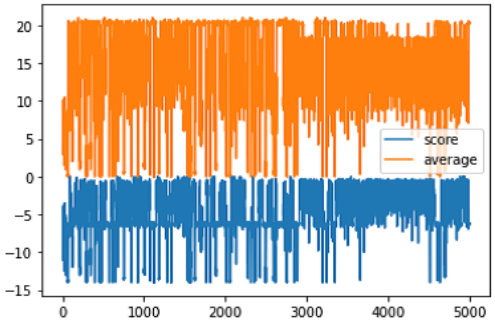
ε -グリーディ法に倣い、学習初期は学習したものではなく乱数を出力に用いる。様々な出力を試行することで、より高いスコアの出力を発見する狙い

がある。このランダムな試行は ε の確率で行われる。 ε の初期値は 1.0 で、学習が行われる (スコア取得 10 回) 毎に 0.05 減算される。

4.1.4　結果

結果、提示される選択肢はゲームを通して同じものばかりになり、それに伴い学習に応じて結果が極端に上下するようになった。ニューラルネットワークでは、入力層への値ひとつが多少変化したところで出力が劇的に変化するような学習は困難であるため、ゲームを通して同じ選択肢ばかりになったものと考察される。

表 3: 一般的な ε -グリーディ法を使用した方法における平均到達階層とスコア



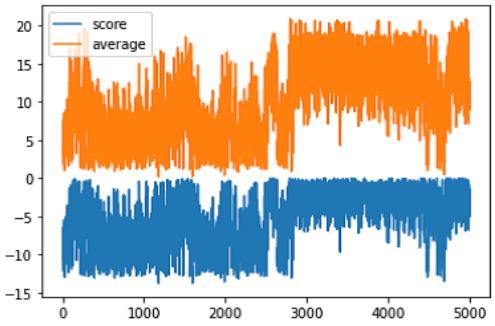
4.2　重み付きランダムを採用した方法

先の方法で取った手法に加えて、選択肢を提示する方法を変更する。出力の最高値 2 つではなく、出力値を重みとしたランダムな抽出で選択肢を提示する方法をとった。

4.2.1　結果

3000 回目前後の学習以降、スコアが安定して-5 以上出る結果となった。選択肢を直接決めるのではなく、何が出やすいのか傾向を決めさせることで、選択毎に変化が生じるゲームとなった。また ε によらない通常の学習でもランダム性を持たせることで、現状よりも良い選択を発見する助けにもなっているのではないかと考察できる。しかし、学習 3000 回目以降も継続して学習を続けても-5 より高いスコアを安定して出すことは叶わなかった。この原因は学習のし過ぎで高いスコアの状態で難易度が安定していないものと仮定された。

表 4: 重み付きランダムを採用した方法における平均到達階層とスコア



4.3 学習に応じて学習率を低下させていく方法

直近のスコアから学習の進み具合を見て、学習率を低下させる方法をとった。先の図において学習 3000 回目以降、目標付近で揺れ動いている average(橙の線) を目標に収束させることが狙いである。

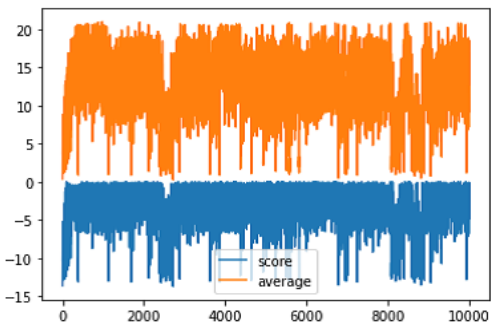
4.3.1 手法

wip

4.3.2 結果

結果、狙い通りにはならず、学習初期より-5 以上のスコアが出るようになるという結果になった。-5 以上のスコアが多く出てはいるが、学習が不足しているのか、しばしば低いスコアが出るという不安定なものとなった。

表 5: 学習に応じて学習率を低下させていく方法における平均到達階層とスコア



4.4　出現する敵を動的に生成する方法

-5 より高いスコアが出ない原因として、一段階易しい選択肢を提示したときに易しすぎ、逆に一段階難しい選択肢を提示したときに難しすぎる、という「帯に短し襷に長し」状態になっているがために振れ幅が大きくなっているという仮説を立てた。これまでは、あらかじめ決められた強さを持つ敵を十数種類から選んでいた。これを変更する。

4.4.1　手法

ほげ

4.4.2　結果

ほげほげ

5　付録

5.1　参考文献

ほげ

5.2　ソースコード

当研究で使用したゲームおよび学習システムのソースコードは、GitHub 上で公開されている。https://github.com/KaichiHamaishi/aiDirectedRPG