電気通信大学大学院　情報理工学研究科
情報・ネットワーク工学専攻　修士論文

「ローグライクゲームにおける大規模ニューラルネットワークを用いた強化学習の研究」

令和 3 年 1 月 25 日

情報数理工学プログラム

学籍番号 1931161

若月裕樹

指導教員　保木邦仁
村松正和

目 次

1　はじめに

ゲームをプレイする人工知能の研究は近年では盛んに行われるようになった．ゲーム人工知能の性能は人工知能研究の達成度の指標となっている．

ゲーム分野においては強化学習法とニューラルネットワークを組合せた手法が目覚ましい成績をあげた．例として 1994 年にバックギャモンで上級者に匹敵する強さを示した TD-Gammon [1] や 2017 年に囲碁でトッププレイヤに勝利した AlphaGo [2] などがあげられる．TD-Gammon は 3 手先読みの探索と TD(λ)，ニューラルネットワークを組合せた手法が使われており，AlphaGo はモンテカルロ木探索と強化学習，深層学習を組合せた手法が使われている．

ごく最近では，より複雑なゲーム，すなわちプレイヤが多人数であったり，不確実性が多かったり，リアルタイム性のあるビデオゲームであったりする場合でも賢い人工知能が開発され始めている [3][4]．

本研究では，最近研究対象として注目を集めつつあるローグライクゲームを題材に強化学習と深層学習を組合せた手法の性能を調査する．ローグライクゲームは 1 人で遊ぶゲームであり，数ターン先の利益を求める短期的な戦略と数百ターン〜数千ターン先の利益を求める長期的な戦略の両方をバランス良くとることが求められる．そのため，プレイヤは様々な選択肢に直面する．このような観点ではローグライクゲームは人工知能開発の難易度が高いゲームであるといえる．

1 人用のビデオゲームの先行研究では特に，Atari 2600 の各種ゲームにおいて Q 学習と深層学習を組合せた DQN と呼ばれる手法が顕著な成果を上げており [5]，十分に熟達した人間プレイヤよりもおおよそ高性能な人工知能が開発されている．

ローグライクゲームの一種であり，本研究で用いた Rogue Clone III と上記のゲームの大まかな特徴の比較を表 1 に示す．ここではゲームプレイの意思決定列をグラフの経路と考えて，このグラフの節点にあたるものを状態，辺にあたるものを行動とみなしている．Rogue Clone III における値は筆者が見積もったゲームクリアした場合の値である．Atari 2600 の各種ゲームと StarCraft II における値は上記の論文 [3][5] より抜粋している．ただし，Atari 2600 は複数のゲームがあるため，行動列長は不明である．状態数が不明としているものは見積もることが困難であるが囲碁よりもさらに多いということは判明している．

これらの特徴は，単純に数値を比較できるものではないかもしれないが，ローグライクゲームは難易度の高いゲームではあるものの強化学習による人工知能の開発が不可能ではないと考えて，本研究ではこのゲームを題材とする．

表 1: ゲームの比較

ゲーム	状態数	行動列長	行動種類数	プレイヤ人数	リアルタイム性
Rogue Clone III	不明	10^4	10^3	1	無
Atari 2600	不明	不明	10^1	1	有
StarCraft II	不明	10^4	10^{26}	2〜	有
囲碁	10^{172}	10^2	10^2	2	無

2 基礎知識

本章では，2.1 節において Rogue Clone III を紹介し，本研究に用いた手法である Q 学習を 2.2 節で，深層学習を 2.3 節で説明する．

2.1 Rogue Clone III

ローグライクゲームの元祖は 1980 年に公開された UNIX 上で動作するゲーム「Rogue」である[1]．Rogue とそれに続く Rogue シリーズは当初はソースコードが公開されておらず，自由にダウンロードして遊ぶことができなかった．Rogue Clone は 同シリーズのファンがそれを再現したゲームであり，ソースコードが公開されている．ローグライクゲームの歴史やソースコードを個人でまとめているウェブサイトによると[2]，現在 Rogue シリーズには様々なバージョンが存在していて，Rogue Clone が主に再現しているものは Rogue 5.3 である．ただし，部分的にバージョン 5.1 のルールが採用されていたり，削除されたゲーム要素があったりするなど，Rogue 5.3 とは若干内容が異なっている．Rogue Clone にもバージョンがあり，最新版が Rogue Clone III である．

このゲームのプレイヤはキーボードでコマンドを入力し，プレイヤキャラクター (以下プレイキャラという) を 2 次元で表示されるダンジョンと呼ばれる空間内で操作する．図 1 は Rogue Clone III のプレイ画面である．プレイ画面は 24×80 の文字で描画される．一番上の行には適宜メッセージが表示され，2〜23 行目にはダンジョン内の現在いる階層のマップが表示される．一番下の行は常にプレイキャラの HP などのステータスが表示されている．

ダンジョンには複数の階層があり，1 つの階層は部屋と通路，罠，隠し通路 (扉) によって構成されている．各階層には様々なアイテムが落ちていたり，様々なモンスターが配置

[1] “Rogue”, Wikipedia, https://en.wikipedia.org/wiki/Rogue_(video_game), (最終アクセス 2021)
[2] yozvox, “Rogue”, http://yozvox.web.fc2.com/526F677565.html, (最終アクセス 2020)

図 1: Rogue Clone III のプレイ画面

されていたりする．これらのオブジェクトは階段を上り下りして，別の階層を訪れるたびにランダムに構成され，基本的にそれぞれ決まった文字でマップに表示される．ただし，プレイキャラの訪れていない場所は表示されず，モンスターは現在いる部屋，もしくは通路の周り 1 マスの範囲までしか表示されない．アイテムは拾うことができ，24 個まで持てる．モンスターは起きているものはマップを移動し，プレイキャラを攻撃する．モンスターを倒すと経験値が手に入る．経験値が溜まるとプレイキャラはレベルが上がり強くなる．

このゲームの目的は

- 「Amulet of Yender」というアイテムを持って 1F の階段を上る (クリアする)
- できるだけ深い階層を目指す
- ハイスコアを狙う

などである．クリア，死亡，宣言して終了のいずれかをするとゲームが終了する．

プレイヤがコマンドを入力しなければゲームは進行せず，コマンドを入力するとまずプレイキャラが行動し，次にそれぞれのモンスターが行動し 1 ターンが経過する．ただし，ターンが経過しないプレイキャラの行動や，何も行動ができずにターンが経過する眠りな

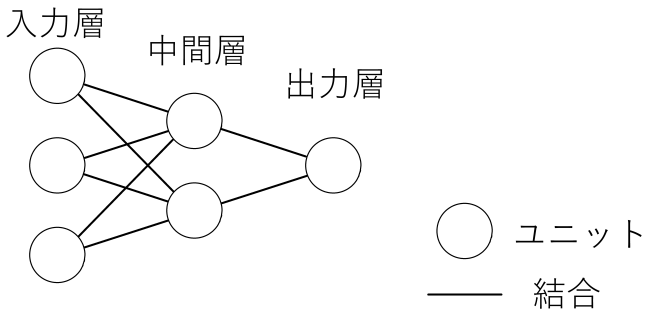

図 2: 3 層 NN の例

どの状態変化がある．ターンが経過するとプレイキャラのお腹が減り，だんだん動けなくなっていき最終的には餓死する．スコアはゲーム終了時に所持している金塊の量である．ただし，クリアした場合は手持ちのアイテムの価値がさらに加算され，死亡した場合はペナルティーとして 1 割引かれる．

アイテムやモンスターなどの詳細な情報は巻末の付録 A にまとめる．

2.2 深層学習

本節では深層学習とそれに関連する知識について，文献 [6] を参考にして記述する．

深層学習とは，ニューラルネットワーク (Neural Network, NN) という数理モデルを用いた機械学習のうち，4 層以上の NN を用いたものである．本論文では順伝播型 NN について説明する．

2.2.1 順伝播型 NN

順伝播型 NN は図 2 のように，ユニットと呼ばれるものが層状に結合した構造を持つ．1 つのユニットは 1 つ以上の実数値を入力として受け取り，それらの総和を活性化関数を通して出力する．ユニットの出力とユニットの入力を繋ぐ結合には重みという実数値が設定されており，ユニットの出力にその重みの値をかけてからユニットの入力に渡す．

図 3 はユニットの入出力の一例である．ユニットの丸の中はそのユニットの出力を表す．図の z は以下のように計算される．

$$u = x_1 w_1 + x_2 w_2 + x_3 w_3 + b \tag{1}$$

$$z = f(u) \tag{2}$$

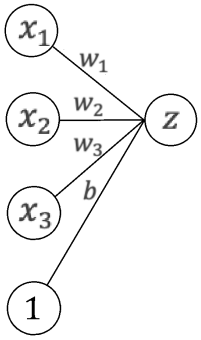

図 3: ユニットの入出力

ここで，b はバイアスと呼ばれる実数値で，f は活性化関数である．

一般化して，第 l 層の i 番目のユニットの出力を $z_i^{(l)}$，第 l 層の i 番目のユニットと $l+1$ 層の j 番目のユニットを結ぶ結合の重みを $w_{ji}^{(l+1)}$，第 l 層の j 個目のユニットに対するバイアスを $b_j^{(l)}$，第 l 層の活性化関数を $f^{(l)}$ とする．このとき，第 $l+1$ 層における j 個目のユニットの出力 $z_j^{(l+1)}$ は以下のようになる．

$$u_j^{(l+1)} = \sum_i z_i^{(l)} w_{ji}^{(l+1)} + b_j^{(l+1)} \tag{3}$$

$$z_j^{(l+1)} = f^{(l+1)}(u_j^{(l+1)}) \tag{4}$$

(3)(4) 式を，重みを行列，出力とバイアスをベクトル形式で表すと以下のようになる．

$$\boldsymbol{u}^{(l+1)} = \boldsymbol{W}^{(l+1)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l+1)} \tag{5}$$

$$\boldsymbol{z}^{(l+1)} = f^{(l+1)}(\boldsymbol{u}^{(l+1)}) \tag{6}$$

ここで，$f^{(l+1)}(\boldsymbol{u}^{(l+1)}) = (f^{(l+1)}(u_1^{(l+1)}), ...)^{\mathrm{T}}$ である．活性化関数は単調非減少の非線形関数が主に用いられる．例をあげると，retified linear (ReL) 関数

$$f(u) = \max(u, 0) \tag{7}$$

である．ここまでは，NN の入力に対する出力の計算を説明した．次節以降では重みを調節する方法を説明する．

2.2.2 誤差関数と勾配降下法

良い NN とは，どんな入力に対しても望みの値を出力できるネットワークである．そのようなネットワークを構築するためには，入力とその入力に対して望ましい値の組 $(\boldsymbol{x}, \boldsymbol{d})$

(以降サンプルという) を複数用いる必要がある．サンプル $(\boldsymbol{x}, \boldsymbol{d})$ の集合を訓練データ D といい，

$$D = \{(\boldsymbol{x}_1, \boldsymbol{d}_1), ..., (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$$

と書く．また，NN の入力 $\boldsymbol{x}$ に対する出力を，全重みとバイアスを成分に持つベクトル $\boldsymbol{w}$ を用いて $\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w})$ と書く．$\boldsymbol{d}$ と $\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w})$ がどの程度異なっているかを測る関数を誤差関数といい，誤差関数の値が小さくなるように $\boldsymbol{w}$ を調節することを NN の学習という．これを最小化することによって，未知のサンプルに対しても $\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w})$ は $\boldsymbol{d}$ の良い推定を与えることが期待される．誤差関数の例に，回帰によく用いられる二乗誤差がある．訓練データ D に対する二乗誤差は以下の式で表される．

$$E(\boldsymbol{w}) = \sum_{k=1}^{|D|} E_n(\boldsymbol{w}) = \sum_{k=1}^{|D|} \frac{1}{2} \|\boldsymbol{d}_k - \boldsymbol{y}(\boldsymbol{x}_k|\boldsymbol{w})\|^2 \tag{8}$$

誤差関数 $E(\boldsymbol{w})$ の最小値を与える $\boldsymbol{w}$ を見つけ出すのは困難である．そのため，極小点を探すことで誤差関数の値が小さくなるように努める．

極小点を探す方法に勾配降下法がある．誤差関数 $E(\boldsymbol{w})$ の勾配は以下のようである．

$$\nabla E(\boldsymbol{w}) \equiv \frac{\partial E}{\partial \boldsymbol{w}} = \left(\frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_M} \right)^{\mathrm{T}} \tag{9}$$

勾配降下法では，現在の $\boldsymbol{w}$ を負の勾配方向に少し動かすことを繰り返す．現在のネットワークパラメータを $\boldsymbol{w}^{(t)}$，動かした後のパラメータを $\boldsymbol{w}^{(t+1)}$ とすると，更新式は以下のようになる．

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \epsilon \nabla E(\boldsymbol{w}^{(t)}) \tag{10}$$

ϵ は学習率と呼ばれる実定数である．初期パラメータ $\boldsymbol{w}^{(1)}$ を適当に決め，(27) の更新式に従ってパラメータを調節していくことで極小点に到達することが多い．ただし，学習率が大きすぎると誤差関数の値が増加してしまうこともあり，小さすぎると学習に時間がかかってしまうこともある．そのため，学習率の値を調節することは学習において極めて重要である．

2.2.3　勾配降下法の改善手法

上記した勾配降下法では，訓練データのサンプル全てを用いて誤差関数や勾配を計算していた．このような学習をバッチ学習という．しかし，この手法には欠点があり，最適解からほど遠い極小解に陥ってしまった場合抜け出せなくなる．

この欠点を改良したものが確率的勾配降下法 (Stochastic Gradient Descent, SGD) である．SGD ではサンプル 1 つ，またはいくつかのサンプルのブロックごとに誤差関数と勾配を決定する．サンプルのブロックをミニバッチという．t 回目のパラメータの更新におけるミニバッチを D_t とすると，誤差関数 $E_t^{\mathrm{mb}}(\boldsymbol{w})$ は以下のように表される．

$$E_t^{\mathrm{mb}}(\boldsymbol{w}) = \frac{1}{|D_t|} \sum_{(\boldsymbol{x},\boldsymbol{d})\in D_t} \frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w})\|^2 \tag{11}$$

サンプルのミニバッチサイズが小さいと SGD の恩恵を強く受けられるが，勾配計算の回数が多いため，計算効率が落ちるという欠点がある．

SGD にはまだ欠点がある．それは誤算関数が谷状の形状になっている場合，パラメータが本来進んでほしい方向へ進むのではなく谷の斜面を振動してしまい，学習の進みが遅くなってしまうということである．この欠点を改善する方法がいくつかあり，その 1 つはモメンタムである．モメンタムは，パラメータの修正量に前回の修正量の定数倍を足すというものである．ミニバッチ $t-1$ に対するパラメータの修正量を $\Delta\boldsymbol{w}^{(t-1)} \equiv \boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)}$ とすると，t 回目のミニバッチに対する更新は以下のようになる．

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \epsilon\nabla E_t^{\mathrm{mb}}\left(\boldsymbol{w}^{(t)}\right) + \mu\Delta\boldsymbol{w}^{(t-1)} \tag{12}$$

μ はどの程度前回のパラメータ修正量を足すかを決定する実定数である．

もう 1 つの改善法に RMSProp がある[3][4]．RMSProp は振動を抑えるために学習率に係数をかけて調節する手法である．RMSProp の t 回目のミニバッチに対する更新は以下のようになる．

$$w_i^{(t+1)} = w_i^{(t)} - \epsilon\left\{\frac{\frac{\partial E_t^{\mathrm{mb}}}{\partial w_i^{(t)}}}{\sqrt{v_i^{(t)} + a}}\right\} \tag{13}$$

ただし，w_i はパラメータ $\boldsymbol{w}$ の要素であり，

$$\begin{aligned} v_i^{(t)} &= \rho v_i^{(t-1)} + (1-\rho)\left\{\frac{\partial E_t^{\mathrm{mb}}}{\partial w_i^{(t)}}\right\}^2 \\ v_i^{(1)} &= 0 \end{aligned}$$

である．ρ はどの程度前回の勾配の大きさを足すかを決定する定数である．a は 0 除算が起こらないためのごく小さい実数値である．$w_i^{(t)}$ が振動している場合，$v_i^{(t)}$ が大きくなり，修正量が抑えられる．

[3] Hinton. G., “Lecture 6.5 - rmsprop, COURSERA: Neural Networks for Machine Learning”.

[4] “Overview of mini-batch gradient descent”, http://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

2.2.4 誤差逆伝播法

誤差逆伝播法は誤差の勾配を効率的に計算するための手法である．これ以降，表記を簡素化するために，+1 を常に出力する特別な第 0 番ユニットを各層に導入し，バイアス $b_j^{(l)}$ を，第 $l-1$ 層の 0 番目のユニットと第 l 層の j 番目のユニットの間の重み $w_{j0}^{(l)}$ として考えることとする．すなわち，第 l 層の j 番目のユニットにおける $u_j^{(l)}$ は以下のようになる．

$$u_j^{(l)} = \sum_{i=1}^{I} w_{ji}^{(l)} z_i^{(l-1)} + b_j^{(l)} = \sum_{i=0}^{I} w_{ji}^{(}l) z_i^{(l)} \tag{14}$$

和の微分は微分の和であるので，バッチ (ミニバッチ) のサイズによらずサンプル 1 つに対する誤差を微分できれば良い．1 つのサンプル n に対する誤差 $E_n(\boldsymbol{w})$ の $w_{ji}^{(l)}$ に関して

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} \tag{15}$$

となる．(15) 式の右辺第 1 項について，$u_j^{(l)}$ の変動が E_n に与える影響は，このユニット j からの出力 z_j^l を通じ，第 $l+1$ 層の各ユニット k の総入力 $u_k^{(l+1)}$ を変化させることによってのみ生じる．したがって，各 $u_k^{(l+1)}$ を経由した微分連鎖により

$$\frac{\partial E_n}{\partial u_j^{(l)}} = \sum_k \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \tag{16}$$

と分解できる．以降，デルタ $\delta_j^{(l)}$ を

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial u_j^{(l)}} \tag{17}$$

と定義する．(16) 式はデルタを使って以下のように書ける．

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} \left(w_{kj}^{(l+1)} f'(u_j^{(l)}) \right) \tag{18}$$

これは $\delta_j^{(l)}$ が $\delta_k^{(l+1)}$ から計算できることを意味する．ここで (15) 式の右辺第 2 項は

$$\frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)} \tag{19}$$

である．したがって，(15) 式は (18) 式 (19) 式を使って，以下のように書ける．

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{20}$$

入力:$(\boldsymbol{x}_n, \boldsymbol{d}_n)$
出力:誤差関数 $E_n(\boldsymbol{w})$ の各層 l のパラメータについての偏微分 $\frac{\partial E_n}{\partial w_{ji}^{(l)}}(l = 2, ..., L)$

1. $\boldsymbol{z}^{(1)} = \boldsymbol{x}_n$ とし，各層 $l(= 2, ..., L)$ の $\boldsymbol{u}^{(l)}$ および $\boldsymbol{z}^{(l)}$ を計算する．
2. 出力層の各デルタ $\delta_j^{(L)}$ を計算する．
3. 中間層 $l(= L-1, ..., 2)$ のデルタを (18) 式に従って計算する．
4. 各層 $l(= 2, ..., L)$ のパラメータ $w_{ji}^{(l)}$ に関する偏微分を (20) 式に従って計算する．

図 4: 誤差逆伝播法のサンプル 1 つに対する勾配計算手順

出力層におけるデルタの計算は誤差関数と活性化関数による．誤差関数が 2 乗誤差，活性化関数が恒等写像であるならば，

$$\begin{aligned}
\delta_j^{(L)} &= \frac{\partial E_n}{\partial u_j^{(L)}} \\
&= \frac{\partial}{\partial u_j^{(L)}} \left(\frac{1}{2} \left\| \boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w}) - \boldsymbol{d} \right\|^2 \right) \\
&= \frac{\partial}{\partial u_j^{(L)}} \left(\frac{1}{2} \left\| \boldsymbol{u}^{(L)} - \boldsymbol{d} \right\|^2 \right) \\
&= u_j^{(L)} - d_j
\end{aligned} \tag{21}$$

である．これらより，誤差逆伝播法のアルゴリズムをまとめると，図 4 のようになる．

2.2.5 NN を構成する層の次元と結合の種類

これまで説明した NN では，層間の全ての結合に対して個別に重みの値が設定されていた．実際には，重みを部分的に共有する結合や特殊な処理をする層がよく用いられる．また，各ユニットの活性化関数は独立した 1 つの層と考えることが多い．NN の実装において，入力や出力を構成する数値列はしばしば 4 次元 (ミニバッチサイズ × チャネル数 × 縦の長さ × 横の長さ) の配列として扱われる．例えば，カラー写真 N 枚の入力は RGB の 3 チャネルを使って，$N \times 3 \times$ 写真の縦の長さ × 写真の横のサイズのように表される．

以下に代表的な NN の層を紹介する．ここで，ミニバッチサイズを 1 とし，入力を $1 \times C \times H \times W$，出力を $1 \times C' \times H' \times W'$ と表す．

全結合層

2.3.1 節で述べた層にあたり，入力側の各結合の重みは共有されない．層の入力 $C \times$

$H \times W$ の値それぞれが異なるユニットに対応している．バイアスを含めたパラメータの数は (入力列長 $+1$)$\times$ 出力列長であるので，$(C \times H \times W + 1) \times C'$ となる．ここで，出力では $H' = 1, W' = 1$ となることとする．

畳み込み層

畳み込み層ではカーネルと呼ばれる二次元配列の重みを使い，出力を計算する．カーネルは $C \times C'$ 個存在するが，次元は $K_h \times K_w$ で統一されている．出力のチャネル m の画像 $H' \times W'$ の格子点 (i, j) の値 u_{ijm} は以下のように計算される．

$$u_{ijm} = \sum_{k=0}^{C-1} \sum_{p=0}^{K_h-1} \sum_{q=0}^{K_w-1} z_{(i+p)(j+q)(k)} h_{pqkm} + b_m \tag{22}$$

z_{ijk} は入力のチャネル k の画像 $H \times W$ の格子点 (i, j) の値，h_{pqkm} は入力チャネル k と出力チャネル m に対応するカーネルの格子点 (p, q) の重みの値，b_m は出力チャネル m のバイアスの値を示す．入力と出力の画像サイズが同じになるように，入力の画像のふちを何らかの値で埋めて入力の画像を大きくするパディングと呼ばれる手法がある．パディングで埋めるふちの幅を P_h，P_w とすると，出力の画像は $H' = H - K_h + 2P_h + 1$，$W' = W - K_w + 2P_w + 1$ となる．パラメータの数は $(C \times K_h \times K_w + 1) \times C'$ となる．

プーリング層

プーリング層ではたたみこみ層と同じようにカーネルを用いるが，重みは存在しない．プーリングにはいくつか種類があるが，最大プーリングの計算は以下のようになる．

$$u_{ijm} = \max_{p \in \{0,\ldots,K_h-1\}} \max_{q \in \{0,\ldots,K_w-1\}} z_{(S_h i+p)(S_w j+q)(m)} \tag{23}$$

ここで S_h は縦のストライド，S_w は横のストライドである．

2.3 Q 学習

本節では Q 学習とそれに関連する知識について，文献 [7][8] を参考にして記述する．

Q 学習は強化学習法の一種であり，1989 年に Chris Watkins によって確立された．強化学習とは，離散的な時間ステップの各々において，意思決定を行うエージェントとエージェントが観測する環境とが相互作用していく中で，エージェントがその経験から目標達成のための何らかの値の見積もりを学習する枠組みである．強化学習にはマルコフ決定過程 (Markov Decision Process, MDP) という数理モデルが用いられる．

2.3.1 MDP

MDP とは上記の相互作用の過程を数理的に表す離散時間の確率制御過程である．MDP は，状態集合 S，行動集合 $A(s)$，初期状態分布 P_0，状態遷移確率 $P(s'|s,a)$，報酬関数 $R(s,a,s')$ で表される．以降，状態集合と行動集合が有限である有限 MDP について記載する．これらの説明は以下の通りである．

状態集合 S

エージェントが観測する環境の様子を状態といい，状態集合は全ての状態からなる有限集合である．状態数が n の状態集合は $S = \{s^1, \cdots, s^n\}$ と表される. また，終端状態が属さない状態の集合を S_c と書く．

行動集合 $A(s)$

エージェントが環境に働きかける作用を行動といい，状態 s の行動集合 $A(s)$ とは，状態 s においてエージェントが選択可能な行動すべての有限集合である．行動数が m の行動集合は $A(s) = \{a^1, \cdots, a^m\}$ と表される．

初期状態分布 P_0

初期状態分布 P_0 とは初期状態 s_0 を決定する S 上の確率分布である．

状態遷移確率 $P(s'|s,a)$

状態遷移確率 $P(s'|s,a)$ とは状態 s でエージェントが行動 a を選択したとき，状態 s' に遷移する確率である．次状態への遷移確率は直前の状態と行動にのみ依存する．

報酬関数 $R(s,a,s')$

状態 s において行動 a を選択し状態 s' となった場合に，エージェントは実数値の報酬 $R(s,a,s')$ を受け取る．

MDP におけるエージェント内のアルゴリズムを図 5 に示す．

エージェントの行動は方策 π によって決定される．方策は任意の状態 s に対して，$A(s)$ 上の確率分布を定める写像である．状態 s において行動 a が選択される確率は $\pi(a|s)$ で表される．

MDP の状態，行動，報酬の列 $s_0a_0s_1r_1a_1...s_Tr_T$ をエピソードと呼ぶ．s_T は終端状態である．例えば，ゲーム人工知能においてはエピソードは 1 回のゲームプレイ，終端状態はゲームクリアかゲームオーバーとなった状態とすることが多い．ただし，終端状態が無くエピソードに分解できない問題もある．

$t \leftarrow 0$
状態 s_0 を観測する
while(状態 s_t が終端状態ではない){
　行動 a_t を取る
　状態 s_{t+1} と報酬 r_{t+1} を観測する
　$t \leftarrow t+1$
}

図 5: MDP におけるエージェント内のアルゴリズム

基本的にある時間ステップ t において，その後の 1 エピソードの報酬の合計 $r_{t+1} + r_{t+2} + ... + r_T$ の期待値を最大化することが強化学習の目的である．ただし，前述のように終端状態が無かったり，エピソードが非常に長かったりする場合のためにここでは，報酬の割引合計

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{24}$$

を最大化することを目的とする．γ は割引率という．$(0 \leq \gamma \leq 1)$

以降はこの報酬の割引合計 G_t を収益，r_t を即時報酬と記述する．

2.3.2 収益最大化とベルマン方程式

収益を最大化するにあたって，収益の期待値を定義する．ある状態 s における方策 π に対する収益の期待値を状態価値と呼び，$V^\pi(s)$ で表す．$V^\pi(s)$ はベルマン方程式

$$\forall s \in S_c, \qquad V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s'|s,a) \left[r(s,a,s') + \gamma V^\pi(s') \right] \tag{25}$$

の解になる．ここで，最も良い方策とは以下の条件を満たす方策 π^* である．

$$\forall s \in S, \qquad V^{\pi^*}(s) = \max_\pi V^\pi(s) \tag{26}$$

π^* を最適方策と呼び，V^{π^*} を特に V^* と書く．V^* に関するベルマン方程式は

$$\forall s \in S_c, \qquad V^*(s) = \max_a \sum_{s'} P(s'|s,a) \left[r(s,a,s') + \gamma V^*(s') \right] \tag{27}$$

となる．π^* は少なくとも一つは存在することが知られている．

ある状態 s において行動 a を選択したのち，方策 π に従ったときの収益の期待値を行動価値と呼び，$Q^\pi(s,a)$ で表す．$Q^\pi(s,a)$ はベルマン方程式

$$\forall s \in S_c, \forall a \in A(s), \qquad Q^\pi(s,a) = \sum_{s'} P(s'|s,a)\left[r(s,a,s') + \gamma V^\pi(s')\right] \tag{28}$$

の解となる．最も良い方策は Q^π を使って書くと，以下の条件を満たす方策 π^* となる．

$$\forall s \in S, \forall a \in A(s), \qquad Q^{\pi^*}(s,a) = \max_\pi Q^\pi(s,a) \tag{29}$$

Q^{π^*} を特に Q^* と書く．Q^* はベルマン方程式

$$\forall s \in S_c, \forall a \in A(s), Q^*(s,a) = \sum_{s'} P(s'|s,a)\left[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \tag{30}$$

の解となる．ベルマン方程式を解くことで最適方策が求まるが，現実には計算資源に限界があるため，これを解くことは困難である．

2.3.3 Q 学習のアルゴリズム

Q 学習は上記の最適行動価値 $Q^*(s,a)$ を求める手法の一つである．Q 学習は方策オフ型 TD 学習法の一種にあたる．方策オフ型というのは学習のために用いられる挙動方策と改善する推定方策が異なるものをいう．Q 学習は行動価値の推定値 $Q(s,a)$ を学習する．

ある時間ステップ t において行動 a_t を選択し，状態 s_t から s_{t+1} に遷移し即時報酬 r_{t+1} を受け取ったとする．このとき，$Q(s_t,a_t)$ は以下の式で更新される．

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t\{r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a) - Q(s_t,a_t)\} \tag{31}$$

α_t はステップサイズパラメータといい，新しく受け取ったデータをどの程度優先させるかの指標になる．図 6 はこの更新の様子を図で表したものである．Greedy 方策とはその時点で最も良いと見積もった行動を確率 1 で選択する方策である．ここで，$a^* \in \arg\max_{a \in A(s_{t+1})} Q(s_{t+1},a)$ である．

挙動方策について，選択されない行動が無い，すなわち

$$\forall s \in S, \forall a \in A(s), \qquad \pi(s,a) > 0 \tag{32}$$

を満たし，さらにステップサイズパラメータについて，

$$\sum_{t=0}^{\infty} \alpha_t = +\infty \tag{33}$$

$$\sum_{t=0}^{\infty} \alpha_t^2 < +\infty \tag{34}$$

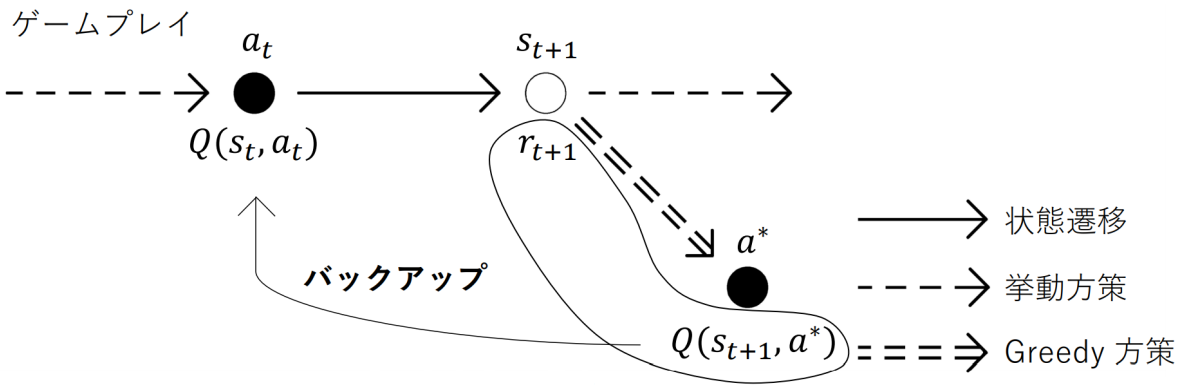

図 6: 行動価値のバックアップ

を満たしているとき，$Q(s,a)$ は $Q^*(s,a)$ に収束することが保証されている．しかし，実践研究の分野では (34) 式の条件は満たさないステップサイズパラメータが用いられることが多い．

2.3.4 関数近似 Q 学習

実際のゲームにおいて状態集合や行動集合は膨大であることが多い．すなわち，$Q(s,a)$ のデータを保持するテーブルが作成できず，かつ全ての状態で選択可能な行動を十分に試行することもできない．このような場合には関数近似を用いた Q 学習を行う．

関数近似を用いる Q 学習では行動価値関数を何らかのパラメータ $\boldsymbol{\theta}$ を用いて表す．状態 s での行動 a の価値は $\hat{Q}(s,a|\boldsymbol{\theta})$ となる．$\boldsymbol{\theta}$ の要素数は M とする．

ある時間ステップ t において行動 a を選択し，状態 s から s' に遷移し即時報酬 r' を受け取ったとする．このとき，$\boldsymbol{\theta}$ を以下の式で更新する．

$$\varepsilon = r' + \gamma \max_{a' \in A(s')} \hat{Q}(s', a'|\boldsymbol{\theta}) - \hat{Q}(s,a|\boldsymbol{\theta}) \tag{35}$$

$$\Delta\boldsymbol{\theta} = \varepsilon \nabla_{\boldsymbol{\theta}} \hat{Q}(s,a|\boldsymbol{\theta}) \tag{36}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha_t \Delta\boldsymbol{\theta} \tag{37}$$

ただし，

$$\nabla_{\boldsymbol{\theta}} \hat{Q} \equiv \frac{\partial \hat{Q}}{\partial \boldsymbol{\theta}} = \left(\frac{\partial \hat{Q}}{\partial \theta_1}, \ldots, \frac{\partial \hat{Q}}{\partial \theta_M} \right)^{\mathrm{T}}$$

である．

関数近似 Q 学習においては (32)(33)(34) 式に加え，挙動方策が変化しないこと，$\hat{Q}(s,a|\boldsymbol{\theta})$ が線形関数であることが収束を保証する条件となる．

ゲーム人工知能の関数近似 Q 学習には深層学習がしばしば用いられる．深層学習を用いる関数近似では近似関数が線形関数ではないため，収束は保証されない．しかし，工夫次第で非常に精度の高い近似関数を構築することが可能となる．

3 先行研究

ローグライクゲームの強化学習の研究は近年になり，多く見られるようになった．ローグライクゲームに関係する先行研究を以下にまとめる．

Rog-O-Matic [9]

1981 年に Carnegie Mellon University の大学院生 4 人が開発した Rogue をプレイする If-Then ルールベースの人工知能である．Rogue におけるプレイヤの目的をいくつかに分け，優先度の高い目的を決めることで，それぞれのエキスパートが具体的な行動を決める．熟達した人間プレイヤのスコアの中央値を上回る性能を持つ．

金川らの研究 [10][11]

簡易化したローグライクゲームのベンチマーク環境「Rogue-Gym」を提案し，さらにそのベンチマーク環境にて DQN の改良アルゴリズム「Double DQN」や「PPO」を用いた実験をした．PPO とは NN を使った Actor-Critic 法の 1 つである．実験に用いたゲームは，本来の Rogue よりもマップが小さく，アイテムが金塊のみで，敵が存在しない．行動も移動に関するもののみである．

加納らの研究 [12][13]

アイテムなどのオブジェクトを省き，マップを小さくして簡易化したローグライクゲームにて以下の 3 手法を組合せた実験を行った．

- A3C… NN を用いた強化学習を並列に行う手法
- ICM… 強化学習の際に用いる報酬をエージェント内部で自動生成する手法
- HRA… 環境の報酬関数をいくつかの報酬関数に分割して，分割されたそれぞれの報酬関数に対してそれぞれ独立したモデルの学習を行う手法

加えて，ICM の改良アルゴリズムである RND と，PPO を用いた実験も行った．

実験に用いたゲームの行動種類数は常に 4 である．階段にたどり着くことを学習の目的としている．

表 2: 先行研究と本研究の成果

研究	手法	ゲームのルール	強化学習
Rog-O-Matic		本来のルール	無
金川らの研究	double DQN, PPO	簡略化	有
加納らの研究	A3C, ICM, HRA, PPO	大幅に簡略化	有
高橋らの研究	モンテカルロ法，パーセプトロンを用いた勝敗予測	少し簡略化	無
本研究	大規模 NN を用いた回帰	本来のルール	有

研究	概略
Rog-O-Matic	熟達した人間プレイヤのスコアを中央値で上回る
金川らの研究	速やかに階段に到達する
加納らの研究	速やかに階段に到達する
高橋らの研究	8 割以上の確率で 5 階に到達する
本研究	If-Then ルール人工知能の収益期待値推定

高橋らの研究 [14]

アイテムと敵の種類を限定して簡易化したローグライクゲームで，If-Then ルール，深さ制限モンテカルロ法および評価関数の教師あり学習を組合せた実験を行った．一手一手の行動が重要な局面ではモンテカルロ法による行動，それ以外では If-Then ルールによる行動を取る方法が良い成績を残した．

4 本研究の目的

本研究では Rogue Clone III のプレイヤが獲得する収益の期待値推定を NN を用いて行う．ここでは，収益推定に関する以下の 2 種類の人工知能の開発を試み，その性能を調査する．

既存人工知能が獲得する収益の期待値を推定する人工知能

ある程度賢い Rogue Clone III をプレイする既存の人工知能が将来に獲得する収益の期待値の推定を行う．NN には，AlphaZero [15] で有効性が示された NN と同程度の規模をもつ NN を用いる．本研究における大規模な NN とは，Rogue Clone III のゲームプレイにおける直近のゲーム状況を全て入力とし，残差 NN (ResNet [16]) と呼ばれる構造を用いる NN のことである．既存の人工知能には，著者が作成した

If-Then ルールベースの人工知能を用いる [17]．この人工知能は Rog-O-Matic を参考に設計され，おおよそ中級者程度の実力を持っている．本研究と先行研究の大まかな特色と主な成果を表 2 にまとめる．

Q 学習を行い，自己訓練を積んだ人工知能

本来のルールのローグライクゲームにおいて，強化学習によってプレイヤの性能が向上したという報告は未だ無い．そこで，本研究では 2020 年現在大規模といわれるような NN を用いると，本来のルールでのプレイヤの強化学習が可能になるかどうかを調査する．強化学習法には Atari 2600 [5] で成果をあげたような Q 学習を用いる．

ここでは 2 つの実験を行う．1 つめの実験では，ローグライクゲームを大幅に簡略化した格子空間のゲームを用いて，Minh らが Atari 2600 で用いた学習法の経験リプレイ並びにターゲットネットワークなどの効果を比較検討し，学習のハイパーパラメータの調節も行う．2 つめの実験では 1 つめの実験で得られた知見をもとに，本来の Rogue Clone III においてプレイヤの Q 学習を行う．

5　実験と考察

この章ではゲームプレイの生成と，実験に用いた手法について説明し，実験結果と考察をまとめる．

5.1　ゲームプレイの生成法

5.1.1　AI-RC3

人工知能がプレイ可能になるように Rogue Clone III を変更して，人工知能用 Rogue Clone III (AI-RC3) を作成した．AI-RC3 では，ゲームプレイは MDP として表現される．変更した主な点を以下に述べる．

通信プロトコル

MDP では状態と行動，即時報酬がエージェントと環境の間でやり取りされる．ここではエージェントは人工知能，環境は AI-RC3 となる．即時報酬は所持している金塊の増減とした (死亡時のペナルティを含む)．これは状態から一意に定まるので，状態と行動のみを通信する．データ通信には POSIX のインターフェイス関数 pipe() などを用いた．

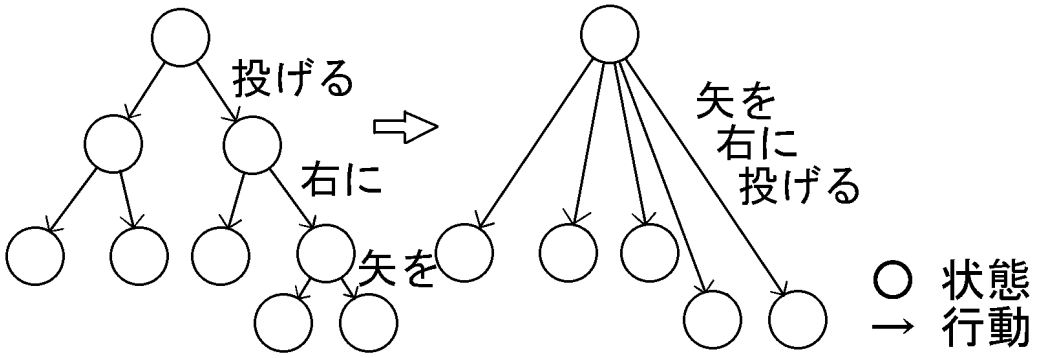

図 7: 行動の指示の単純化

状態の構成

通信される状態の情報は，マップ，メッセージ，ステータス，アイテム情報で構成される．マップ情報とステータス情報は文字が様々な順番で組合された列なので，これらを簡潔に表すことは難しい．一方で，メッセージ情報やアイテム情報は英単語の列で表されるため，より簡潔に表すことが可能である．本実験では，メッセージやアイテム名には 1 対 1 に対応するインデックスを付けた．ここで，効果が判明していないアイテムと判明したアイテムとのインデックスの対応付けは，ゲームを開始する毎に変わるようにした．アイテムとメッセージのインデックスの対応表は巻末の付録 B に示す．

行動の構成

本来の Rogue Clone III では，行動の決定は，人間プレイヤが指示を階層的に行うことでなされる．これを図 15 のように 1 回で全ての行動を選択できるように変更した．行動も 1 対 1 に対応するインデックスを付けた．そして，巻物を読む (r) や水薬を飲む (q) といったコマンドは対象となるアイテムが被ることが無いので，これらはアイテムを使うという行動にまとめた．また，明らかに人間プレイヤ用である行動は削除した．例えば，直近のメッセージを読み直す行動や，壁にぶつかるまで直線に進む行動が該当する．これらの結果，行動集合の大きさは最大で 462 となった．行動のインデックスの対応表も巻末の付録 B に示す．

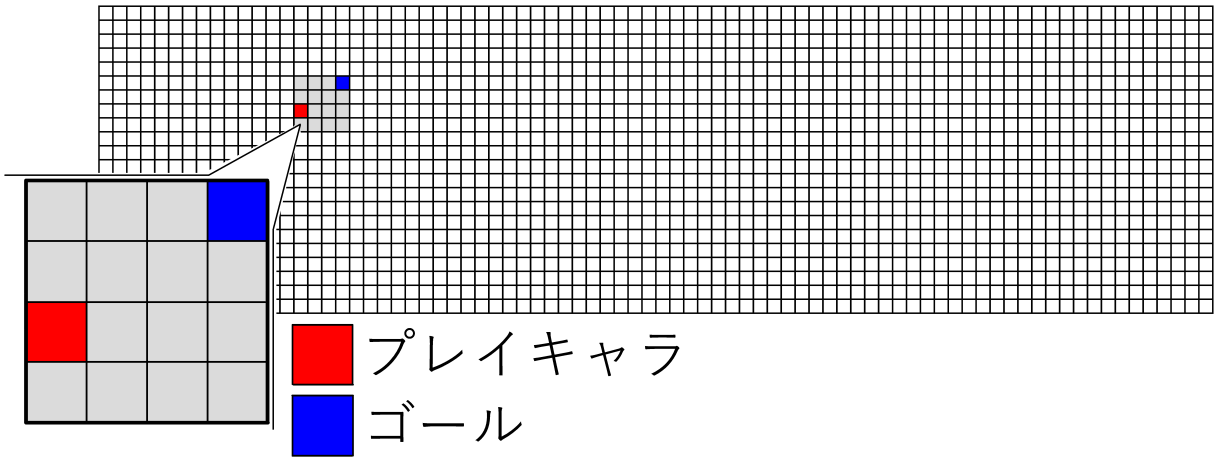

図 8: 格子空間のゲーム

5.1.2 格子空間のゲーム

格子空間のゲームは Rogue Clone III と同じ移動ルールを持つが，アイテムやメッセージ，ステータスは無く，敵もいない (図 8 参照)．四角形の部屋が 22×80 の二次元格子空間の中のランダムな位置に 1 つ配置され，その部屋の中にプレイキャラとゴールが重ならないようにランダムに配置される．部屋の外に出ることはできない．キャラクターがゴールの上に来ればゲームクリアであり，できるだけ早くゴールを目指すゲームとなっている．

行動集合の大きさは 21 で，移動行動 16 種類に Rogue で待機や探索にあたる行動 (このゲームでは意味がない) などを足したものが格子空間のゲームでの行動である．プレイヤの受け取る即時報酬は常に -1 である．

ここでは 2 種の格子空間ゲームを扱い，一方は部屋の大きさが 2×2，他方は 4×4 である．

5.2 実験に用いた学習アルゴリズム

この節ではまず Q 学習のアルゴリズムから説明する．図 9 は最も単純なゲームプレイヤの深層 Q 学習の基本アルゴリズムである．この基本アルゴリズムの性能を向上させるための発展手法がいくつかあり，以下に本実験で用いたものを示す．

経験リプレイ

図 9 のアルゴリズムでは状態遷移を観測した後 (6 行目)，すぐにそのデータを用いて NN を学習する (9 行目)．しかし，これではデータの並びに規則性ができてしまい，SGD などが損失関数値を最小化することに失敗することが多々ある．そのため，

```
while (true){
  ゲーム開始
  ゲームの初期状態 s を観測する
  while (s が終端状態ではない){
    挙動方策に従って行動 a を取る
    新しい状態 s' と即時報酬 r' を観測する
    s' において，NN が推定する行動価値が最大の行動 a* を選ぶ
    組 (s, a, r', Q̂(s', a*)) を元に訓練データのサンプルを作成する
                                        ※ (35) 式参照
    損失関数の値が小さくなるように NN のパラメータを 1 回更新する
                                        ※ (12)(36)(37) 式参照
    s ← s'
  }
}
```

図 9: 深層 Q 学習の基本形

1: リプレイメモリを空にする
2: while (true){
3: ゲーム開始
4: ゲームの初期状態 s を観測する
5: while (s が終端状態ではない){
6: 挙動方策に従って行動 a を取る
7: 新しい状態 s' と即時報酬 r' を観測する
8: 組 (s, a, s', r') をリプレイメモリに追加する
9: if (リプレイメモリが一定長に達している){
10: リプレイメモリからミニバッチサイズ分の組 $(\bar{s}, \bar{a}, \bar{s'}, \bar{r'})$ を無作為に選択してコピーし，古い組をミニバッチサイズ分削除する．
各 $(\bar{s}, \bar{a}, \bar{s'}, \bar{r'})$ に対して
11: $\bar{s'}$ において，NN が推定する行動価値が最大の行動 $\bar{a^*}$ を選ぶ
12: 組 $(\bar{s}, \bar{a}, \bar{r'}, \hat{Q}(\bar{s'}, \bar{a^*}))$ を元に訓練データのサンプルを作成する
※ (35) 式参照
13: 損失関数の値が小さくなるように NN のパラメータを 1 回更新する
※ (12)(36)(37) 式参照
14: }
15: $s \leftarrow s'$
16: }
17: }

図 10: 経験リプレイを利用した深層 Q 学習の基本形

1: リプレイメモリを空にする
2: while (true){
3: ゲーム開始
4: ゲームの初期状態 s を観測する
5: while (s が終端状態ではない){
6: 挙動方策に従って行動 a を取る
7: 新しい状態 s' と即時報酬 r' を観測する
8: s' における，NN が推定する行動価値が最大の行動 a^* を選ぶ
9: 組 $(s, a, r', \hat{Q}(s', a^*))$ を元に訓練データのサンプルを作成する
※ (35) 式参照
10: サンプルをリプレイメモリに保存する
11: if (リプレイメモリが一定長に達している){
12: リプレイメモリからミニバッチサイズ分のサンプルを無作為に選択してコピーし，古いサンプルをミニバッチサイズ分削除する．
13: 損失関数の値が小さくなるように NN のパラメータを 1 回更新する
※ (12)(36)(37) 式参照
14: }
15: $s \leftarrow s'$
16: }
17: }

図 11: 目標値を事前計算する深層 Q 学習の基本形

表 3: 既存人工知能の性能

平均スコア (G_0)	平均行動列長	平均到達階	平均獲得経験値
172 ± 5	910 ± 18	3.67 ± 0.04	56.7 ± 2.1

データの並びにランダム性を持たせる必要がある．そこで図 10 のような経験リプレイを利用した手法を用いる．この手法では状態・行動・即時報酬列をある程度溜めておき (8 行目)，一定長溜まった後にランダムに 1 ステップ分の遷移を取り出して NN を学習する (13 行目)．

ターゲットネットワーク

訓練データの目標値 (ターゲット) を作成する際に，NN が頻繁に変わることは良くないとされている．そこでターゲットネットワークというものを頻繁に更新する NN とは別に用意しておき，一定回数 NN の更新を行ってから，ターゲットネットワークに NN をコピーする．アルゴリズムは経験リプレイを用いる手法であれば，図 10 の 11 行目の NN をターゲットネットワークに変更したものである．

目標値事前計算手法

経験リプレイを用いる際，リプレイメモリに状態遷移を登録する前にその時点での NN で目標値まで計算し，リプレイメモリに登録するという手法である．ターゲットネットワークと同様に新しすぎない NN で目標値を計算する上，急激に目標値を出す NN が変化することがない．図 11 はこの手法のアルゴリズムである．

次に収益推定のアルゴリズムを説明する．図 12 は既存人工知能が状態 s_t で行動 a_t を選択した後にゲームが終了するまでに獲得する収益 G_t の期待値を推定する回帰のアルゴリズムである．ただし，収益の計算における割引率 γ は 1.0 である．ここでプレイヤの行動の決定には著者が Rog-O-Matic を参考に作成した If-Then ルールベースの人工知能を用いた．この人工知能の性能を表 3 に示す．これらの値はゲームプレイ 20000 回の平均と，標準誤差の 2 倍から計算されたおおよその 95%信頼区間を表す．

5.3 NN の構築と入力の特徴

本研究では 3 種類の NN を用いた．それぞれを大規模 NN，中規模 NN，小規模 NN と命名した．小規模 NN は AI-RC3 のプレイヤには用いることができないが，格子空間のゲームでの実験において実験時間の短縮のために用いた．

1: リプレイメモリを空にする
2: while (true){
3: ゲーム開始
4: $t \leftarrow 0$
5: ゲームの初期状態 s_0 を観測する
6: while (s_t が終端状態ではない){
7: 挙動方策に従って行動 a_t を取る
8: 新しい状態 s_{t+1} を観測する
9: $t \leftarrow t+1$
10: }
11: $0 \leq t' \leq t-1$ の各 t' において，
組 $(s_{t'}, a_{t'}, G_t)$ をリプレイメモリに保存する
12: while (リプレイメモリが一定長に達している){
13: リプレイメモリからミニバッチサイズ分の組 (s, a, G_t) を
無作為に選択してコピーし，古い組をミニバッチサイズ分削除する．
14: 損失関数の値が小さくなるように NN のパラメータを 1 回更新する
※ (12) 式参照
15: }
16: }

図 12: 経験リプレイを用いた回帰のアルゴリズム

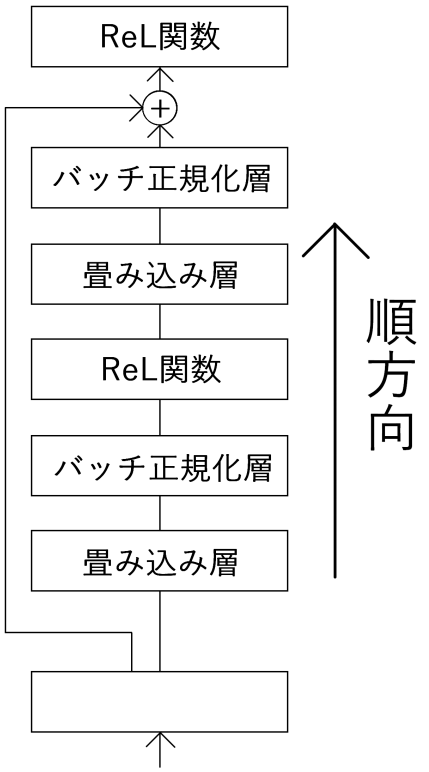

図 13: ResNet のブロック 1 つ分の構造．$\oplus$ の部分は 2 つの入力の値を成分ごとに足し合わせる処理をしている．

はじめに大規模 NN の構造を説明する．通常，深層学習における畳み込み層の数が多すぎると勾配計算における勾配消失/勾配爆発が起きやすく，安定した学習は困難とされている．そして，残差 NN (ResNet) と呼ばれるネットワーク構造を使うとこのような問題が軽減されると考えられている．ResNet は，いくつかの層が 1 つのブロックを形成し，これがいくつも重なっている NN である．本実験ではこのブロックは畳み込み層 2 層，バッチ正規化層 [18]2 層，ReL 層 2 層からなる (図 13 参照)．NN は深ければ深いほどより表現力が増すことが経験上知られており，この構造を大規模 NN に用いた．大規模 NN の構造は図 14 の様である．この NN の deconv 層は次元 1×1 の画像を 11×20 の画像に，同じ値をコピーして拡張し出力する．ResBlock は図 13 のブロック 1 つである．層の間に記載されている数値はミニバッチサイズが 1 のときの出力の次元である．NN の出力数は AI-RC3 の行動集合の大きさと同じ数であり，それぞれの行動に対応する値が出力される．なお，格子空間のゲームでも NN の出力数は同じだが，このゲームに存在しない行動については出力の値は無視される．

次に，大規模 NN の入力について説明する．AI-RC3 と格子空間のゲームでは入力が異

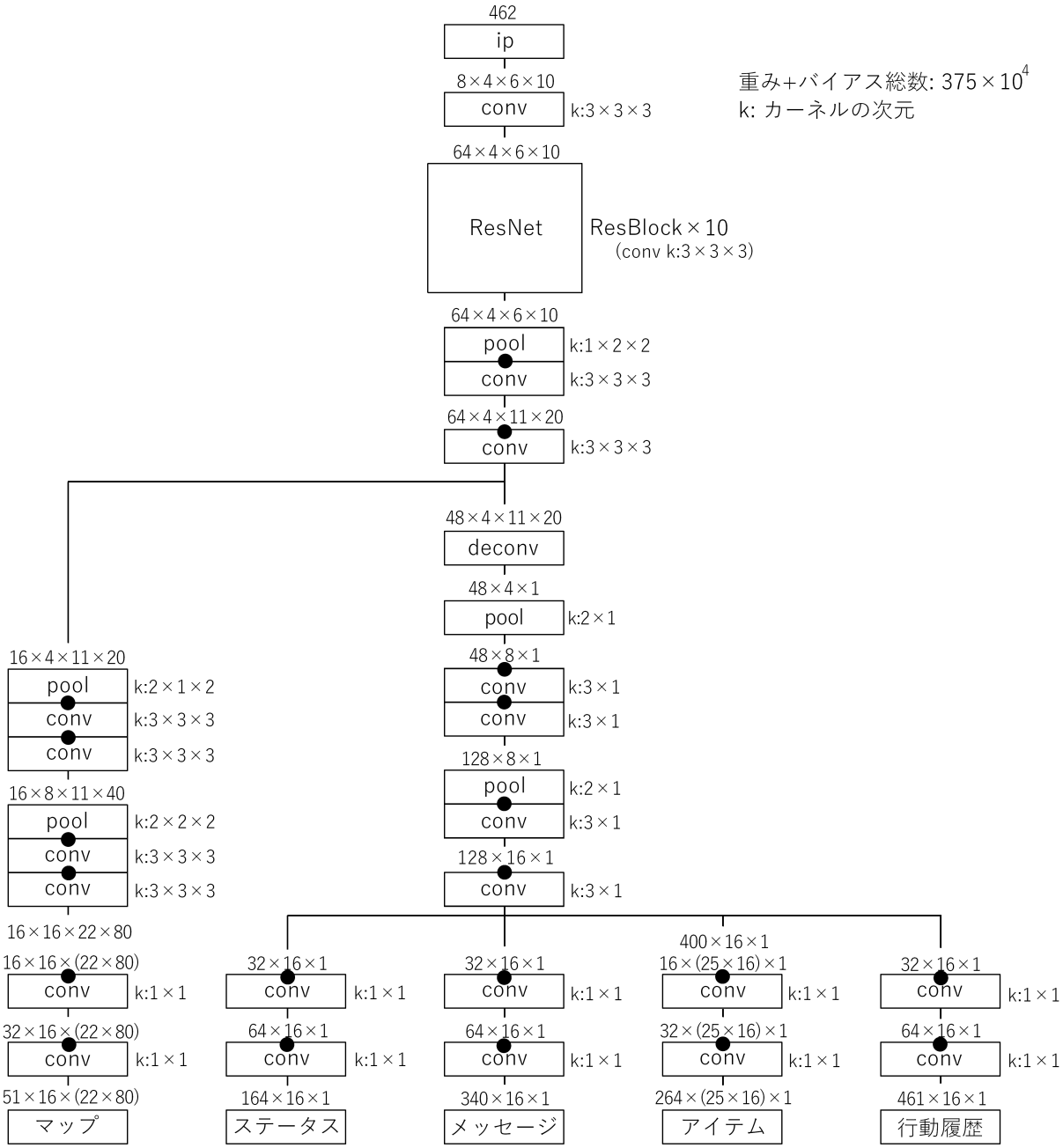

図 14: 大規模 NN の構造．conv は畳み込み層，pool はプーリング層，ip は全結合層を示す．また●はバッチ正規化層と ReL 層が付いていることを表す

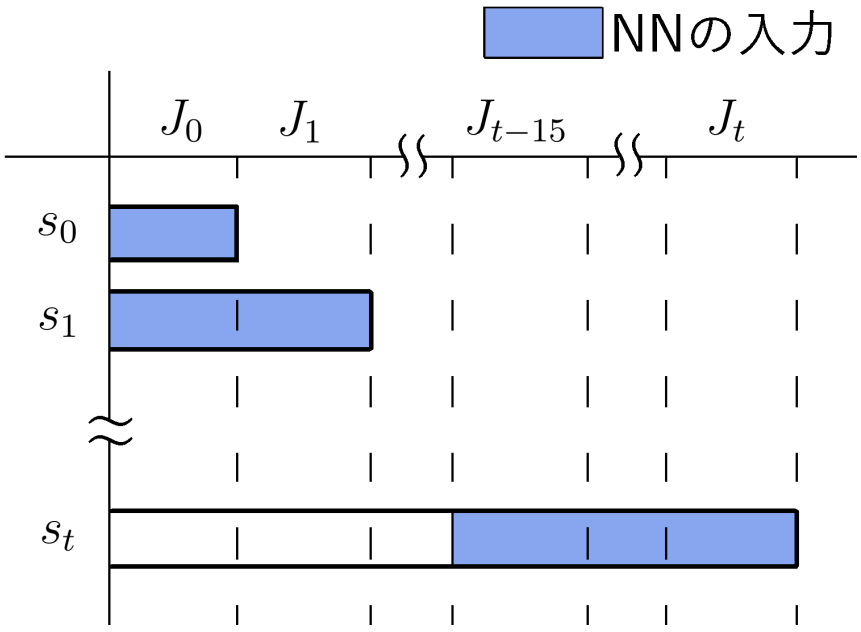

図 15: 状態と大規模 NN の入力の対応

なり，まず AI-RC3 の入力について説明する．本来 Rogue Clone III ではゲーム開始時からのメッセージやコマンドの履歴などの情報が状態に対応するものである．しかし，NN の計算などの都合上，これらの情報を全て符号化して NN に与えることは難しい．よって，ある状態 s_t の大規模 NN の入力は図 15 のようにした．ここで J_t はタイムステップ t において AI-RC3 から送られてくるマップなどの情報である．具体的な NN の入力は直近 16 ステップ分の AI-RC3 から送られてくる情報の組 (マップ，メッセージ，ステータス，アイテム) を表す数値列，そして直近 15 ステップ分の行動履歴を表す数値列とした．大規模 NN に入力する特徴と図 14 の入力層との対応を以下に説明する．

マップ

入力の配列は $51 \times 16 \times (22 \times 80)$ である．ここで，22×80 はマップの大きさに対応する．16 は用いる情報列 $\{J_t\}$ の長さ．51 はマップの特徴を表すチャネルで，それぞれの内訳は表 4 に示す．それぞれのチャネルは該当するものがある場所の値が 1，それ以外が 0 となる．

ステータス

入力の配列は $164 \times 16 \times 1$ である．ここで，16 は用いる情報列 $\{J_t\}$ の長さ．164 はステータスの特徴を表すチャネルで，それぞれの内訳は表 5 に示す．

表 4: マップの特徴

チャネル	意味
0	プレイキャラに対応するチャネル
1〜7	地形 7 種類に対応するチャネル
8〜16	落ちているアイテム 9 種類に対応するチャネル
17〜42	モンスター 26 種類に対応するチャネル
43〜46	炎/氷を当てた/かわしたモンスターのチャネル
47	凍り付いたモンスターのチャネル
48	炎を吐いたモンスターのチャネル
49	攻撃を当ててきた可能性のあるモンスターのチャネル
50	攻撃を外してきた可能性のあるモンスターのチャネル

表 5: ステータスの特徴

チャネル	意味
0〜98	今いる階層-1 のチャネルのみ 1 になり，他は 0 になる
99	$\log_{10}$(金塊の量 + 1)/5 の値になる
100	$\log_{10}$(現在 Hp)/2 の値になる
101	$\log_{10}$(最大 Hp)/2 の値になる
102〜128	チャネル 102〜チャネル (98+現在 Str) までが 1 となり，他は 0 となる
129〜155	チャネル 129〜チャネル (125+最大 Str) までが 1 となり，他は 0 となる
156	Arm/10 の値になる
157	プレイキャラのレベル/20 の値になる
158	$\log_{10}$(経験値 + 1)/7 の値になる
159〜162	お腹の好き具合 (normal:159,hungry:160,...) の値が 1 になる
163	$\log_{10}$(乗った金塊の量)/2 の値になる

表 6: メッセージの特徴

チャネル	意味
0〜99	(来たメッセージの インデックス − 2) のチャネルが 1 になる
100〜339	1 ステップで複数回来る可能性があるメッセージ 16 種について 最大 16 回までカウントし，来ただけ対応するチャネルが 1 になる

表 7: アイテムの特徴

チャネル	意味
0〜115	そのアイテムの種別の インデックス − 1 のチャネルが 1 になる
116〜214	チャネル 116〜チャネル (115+そのアイテムの所持数) までが 1 となる
215	効力の分かっている武器/防具なら 1 になるチャネル
216	装備中のアイテムなら 1 になるチャネル
217〜225	チャネル 217〜チャネル (219+武器の命中補正値) までが 1 となる． 効力未判別ならば補正値 0 と仮定
226〜234	チャネル 226〜チャネル (228+武器の攻撃補正値) までが 1 となる． 効力未判別ならば補正値 0 と仮定
235〜249	チャネル 235〜チャネル (240+鎧の補正値) までが 1 となる． 効力未判別ならば補正値 0 と仮定
250	錆除けされている鎧ならば 1 となるチャネル
251〜256	チャネル 251〜チャネル (253+指輪の補正値) までが 1 となる． 効力未判別ならば補正値 0 と仮定
257〜263	チャネル 257〜チャネル (256+杖の残り回数) までが 1 となる． 効力未判別ならば補正値 0 と仮定

メッセージ

入力の配列は $340 \times 16 \times 1$ である．ここで，16 は用いる情報列 $\{J_t\}$ の長さ．340 はメッセージの特徴を表すチャネルで，それぞれの内訳は表 6 に示す．

アイテム

入力の配列は $264 \times (25 \times\ 16) \times 1$ である．ここで，16 は用いる情報列 $\{J_t\}$ の長さであり，25 は所持アイテムの上限 +1 (1 チャネルは踏んだアイテムの情報を入れるために用意) である．264 はアイテム 1 つの特徴を表すチャネルで，それぞれの内訳は表 7 に示す．

行動履歴

入力の配列は $461 \times 16 \times 1$ である．ここで，16 は他の入力に合わせて 16 としているが，行動の履歴は 15 ステップ分のみ入力される．461 は行動集合の大きさから 1 を引いたものである．取った行動の インデックス -1 のチャネルが 1 になり，他は 0 となる．インデックス 0 の行動はゲーム状態を確実に終端させる行動なので NN に入力することを考えない．

また，タイムステップ t が 15 未満では入力する値を補完するために J_0 に対応する入力の値をコピーして NN の入力を全て埋めている．

一方で，格子空間のゲームではステータスやメッセージ，アイテムの情報は存在しないため，これらに相当する入力の値は 0 とした．マップの入力も 3 チャネル (プレイヤキャラの情報, ゴールの情報, 部屋の情報) のみ使用し，他のチャネルの入力の値は 0 である．

次に中規模 NN の説明をする．中規模 NN の構造は図 14 の ResNet 部分を畳み込み層 2 層分 (ReL 層などを含む) に変換したものである．中規模 NN の入力は大規模 NN と同等である．

最後に小規模 NN の説明をする．小規模 NN の構造は図 16 の様である．この小規模 NN の入力は格子空間のゲームの 1 つのタイムステップにおけるマップ情報である．マップ情報は 3 チャネル (プレイヤキャラの情報，ゴールの情報，部屋の情報) で表される．なお，NN の出力数は 438 となっているが，これは著者が誤って AI-RC3 の行動のうち，アイテムの鑑定をする行動 24 種を含めなかったためである．先述の通り，この NN は AI-RC3 では使用しないため，これが実験結果に与える影響は限定的なものであろう．

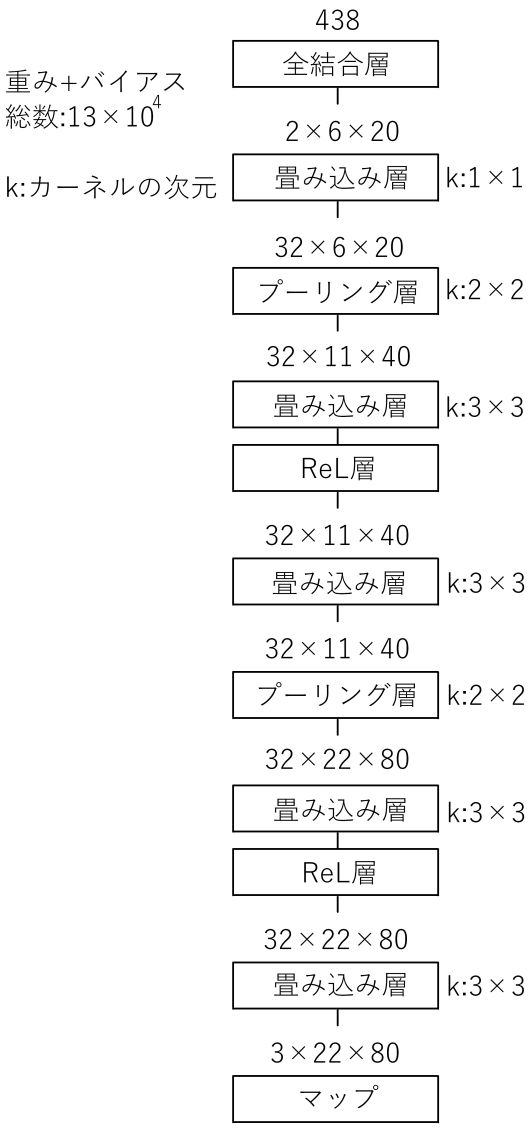

図 16: 小規模 NN の構造

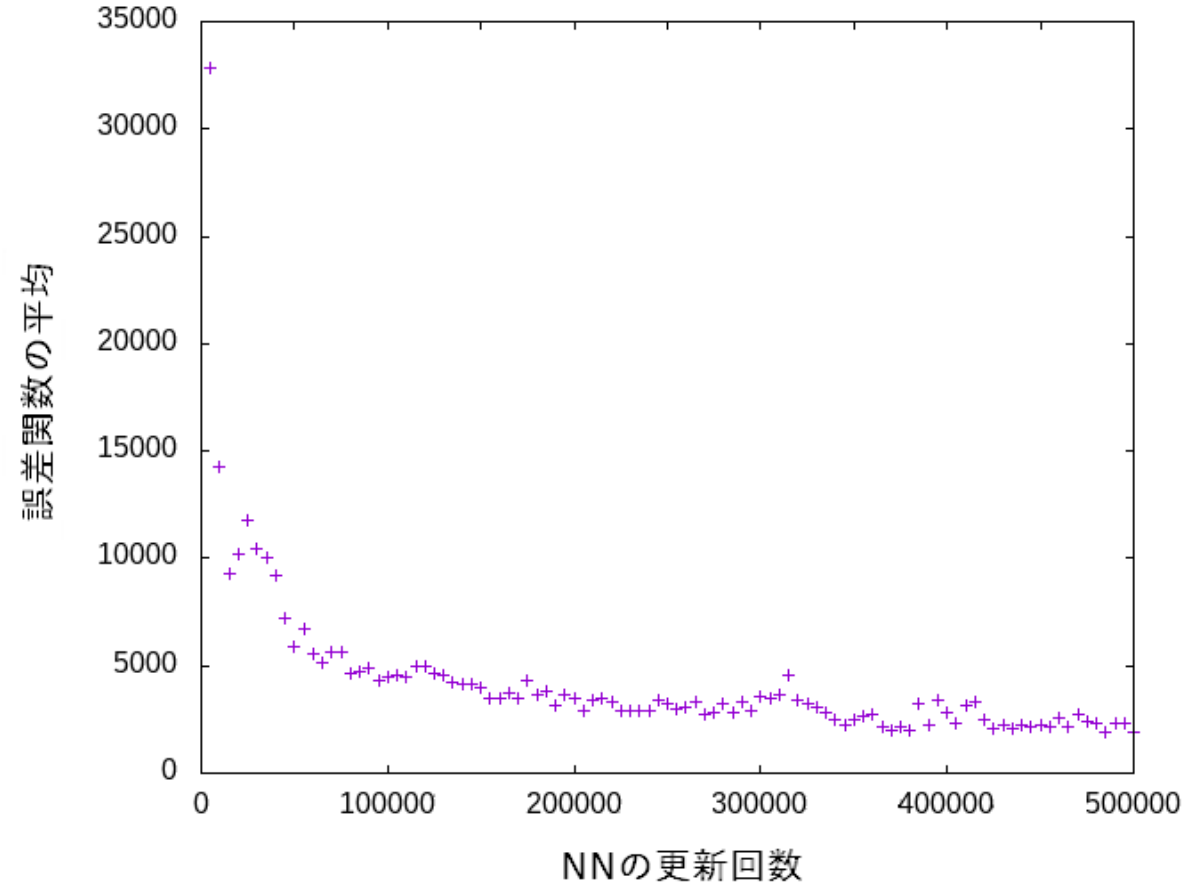

図 17: 収益推定の 2 乗誤差

5.4 実験結果と考察

以下の実験は Caffe 1.0 を用いて行った．また学習の最適化アルゴリズムには RMSProp を用いた．

5.4.1 既存人工知能が獲得する収益の期待値の推定

図 12 のアルゴリズムを用いて，AI-RC3 における既存人工知能の獲得する収益を推定する大規模な NN の学習を行った．図 12 のアルゴリズムにおいて，5, 7, 8 行目で AI-RC3 と通信を行う．また，14 行目の NN は図 14 の大規模 NN である．学習率は 0.0001，ミニバッチのサイズは 64，リプレイメモリのサイズは状態遷移 15×10^5 個とした．

図 17 の横軸は学習における NN の更新回数であり，縦軸は NN の推定値と収益の 2 乗誤差のミニバッチ学習 5000 回ごとの平均値である．学習が進むに連れておおよそ誤差が

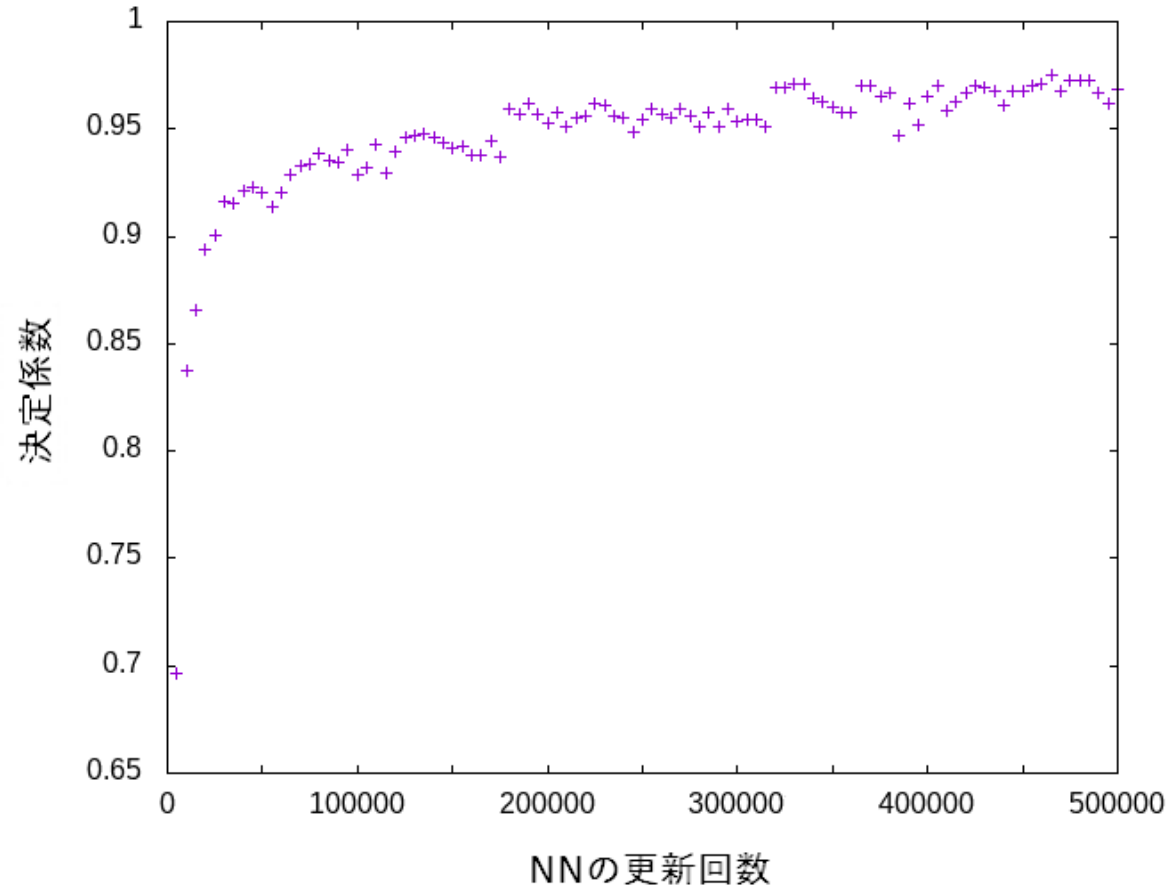

図 18: 収益推定の決定係数

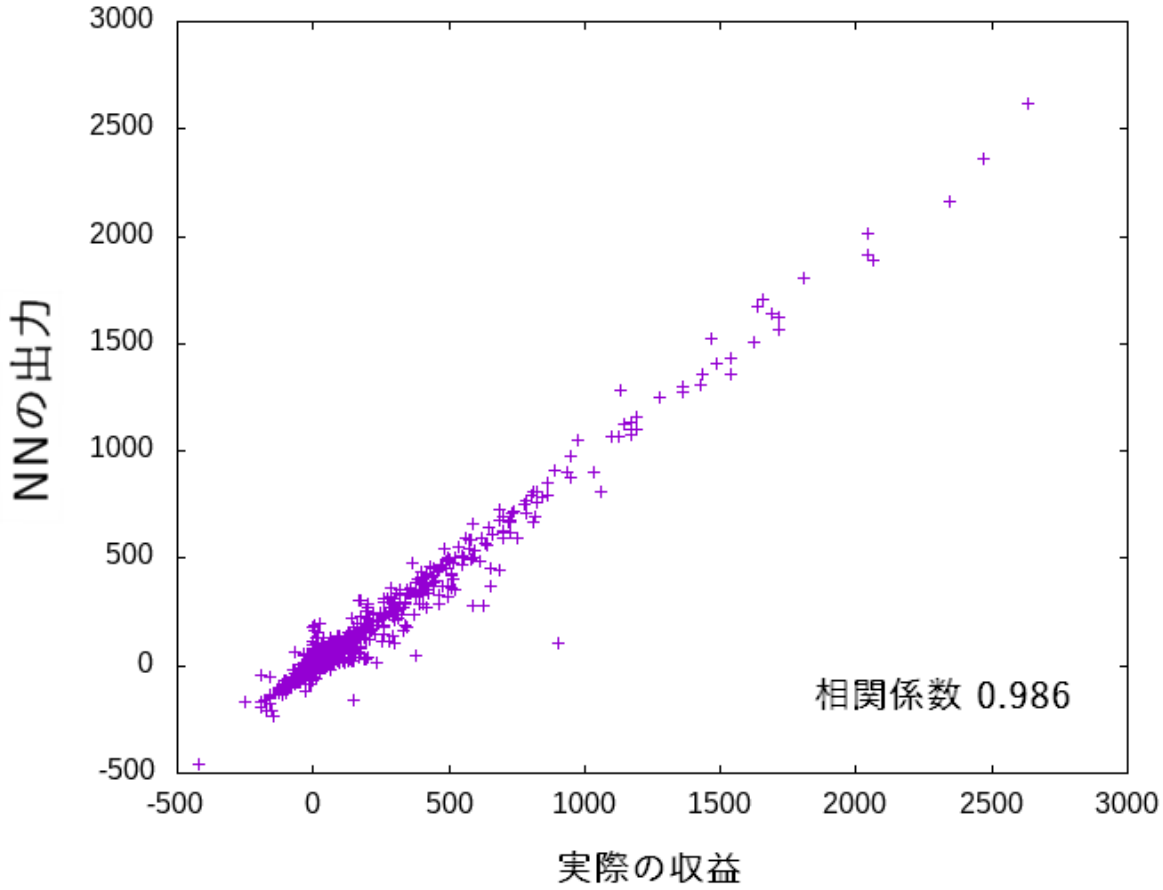

図 19: 収益推定の実際の収益と推定値の散布図

減少していることがわかる．

性能を計測するために決定係数を用いる[5]．図 18 の横軸は学習における NN の更新回数であり，縦軸はミニバッチ学習 5000 回ごとの決定係数である．おおよそ 0.97 程度に達しており，十分な推定の精度が得られたといえる．

また，図 19 は既存人工知能が獲得した収益と 500000 回ミニバッチ学習をした後の NN の推定値の散布図である．横軸は収益であり，縦軸は NN の推定値である．相関係数は 0.986 になった．

総合して，非常に高い精度が得られたといえるが，これは既存人工知能の思考が影響している点もあると考えられる．例えば，この人工知能は隠し扉を見つける能力が低く，対処できない階層の構造だと階段を発見できないまま餓死を待つことしかできなくなってしまう．この場合，収益の予測は容易であり，餓死までのステップ数が多いため，学習のサンプルにこのような状況が多く含まれることになる．従って，既存人工知能の収益期待値の推定は、人間プレイヤのそれの推定よりも容易だったのではないかと考えられる．

また結果は示さないが，中規模 NN で同様の実験をしたところ，ほぼ同様の結果が得られた．既存人工知能の収益推定の精度は，大規模と中規模 NN 共に決定係数 0.97 程度であり，良好であった．

5.4.2 Q 学習による自己訓練

図 9 のアルゴリズムやその発展形を用いて，Q 学習による自己訓練を行った．本節で示す実験は，次の 2 つである．1 つめの実験では，格子空間のゲームを用いて，経験リプレイ並びにターゲットネットワークなどの効果を比較検討し，学習のハイパーパラメータの調節を行った．そして，いくつかの手法やハイパーパラメータについて著者が得た知見を述べる．2 つめの実験では 1 つめの実験で得られた知見をもとに，AI-RC3 においてプレイヤの Q 学習を行う．両方の実験において，挙動方策は ε−Greedy 法を用いた．これは確率 ε で一様ランダムに行動を取り，確率 $1-\varepsilon$ で NN が最も良いと推定する行動を取る方策である．

図 9, 10, 11 のアルゴリズムや種々の規模の NN を，格子空間のゲームに適用し得られた知見を次にまとめる．ここで学習がうまくいくというのは，NN の誤差が減少するだけでなく，状態に応じて行動の優劣が正しく付けられるようになることを意味する．

経験リプレイ

部屋の大きさが 4×4，位置が固定の格子空間のゲームにおいて，経験リプレイを用

[5]決定係数の定義には，文献 [19] の式 (1) を用いた．

いなかった場合，どの手法の組合せも学習がうまく進まなかった．小規模な NN を使い，経験リプレイを用いた場合は非常にスムーズに学習が進んだ．これらの結果から経験リプレイは特に有用であるといえる．以降の実験は全て経験リプレイの手法を用いている．

ResNet

ResNet を用いた大規模な NN を様々な格子空間のゲームの学習に用いたがどれも学習が進まなかった．中規模 NN を用いた結果，2×2 や 4×4 の格子空間で学習が進むようになったため，ResNet は本実験では有効では無いことが分かった．

ターゲットネットワークと目標値事前計算手法の比較

ターゲットネットワークと目標値事前計算手法の比較を表 8 に示す．ターゲットネットワークの更新頻度は 1, 2 行目のものがミニバッチ学習 20000 回ごと，3 行目のものがミニバッチ学習 1000 回ごとである．格子空間のゲームプレイヤの Q 学習全体の傾向として，学習がうまく進むときはしばらく誤差関数の値が上昇を続けた後，減少に転じるという特徴があった (表 8 の 6 行目の実験における誤差関数の値の推移を図 20 に示す)．そのため，収束の早さの指標として誤差が減少傾向に転じる NN の更新回数を記載している．この値は小さいほど良い値である．小規模な NN では，ターゲットネットワークを用いる方が目標値を事前計算する手法よりも成績が良かった．しかし，中規模な NN ではターゲットネットワークは性能が振るわなかった．これらの結果から，中規模以上の NN ではターゲットネットワークは効果が薄い，もしくはハイパーパラメータの調節が困難であると予想される．また，ターゲットネットワークも目標値の事前計算も用いない場合，極端に学習の効率が悪くなるか，学習に失敗した．そのため，いずれかの手法を学習に用いた方がいいと考えられる．

ε−Greedy 法

表 9 は ε に関する比較である．挙動方策に用いた ε−Greedy 法について，小規模な NN を用いる学習や 2×2 の部屋での学習においては ε が 1 の方が学習が早く進んだ．しかし，4×4 の部屋のゲームで中規模 NN を用いたもののみ一様ランダムに行動する挙動方策の方が極端に収束が遅くなった．ε の値が大きければ大きいほど探索行動を取るが，中規模以上の NN で学習をするときは一様ランダムの挙動方策よりも，ある程度 Greedy な行動を取るもののほうが良いと考えられる．

以上で得られた知見を参考にし，AI-RC3 において，大規模と中規模 NN を用いてプレイヤの Q 学習を行った．学習率は 0.00006，ミニバッチサイズは 64，リプレイメモリのサ

表 8: 目標値計算に関する手法の比較

手法	部屋	NN	ε	バッチサイズ	学習率	誤差の減少開始 (NN 更新回数)
ターゲットネットワーク	2×2	中規模	0.8	64	0.000001	140000
ターゲットネットワーク	4×4	中規模	0.8	64	0.000001	減少確認できず
ターゲットネットワーク	4×4	小規模	0.8	32	0.0003	27000
目標値の事前計算	2×2	中規模	0.8	64	0.00003	14000
目標値の事前計算	4×4	中規模	0.8	64	0.00001	54000
目標値の事前計算	4×4	小規模	0.8	32	0.0003	73000

表 9: $\varepsilon-$Greedy 法の比較

手法	部屋	NN	ε	バッチサイズ	学習率	誤差の減少開始 (NN 更新回数)
目標値の事前計算	4×4	中規模	0.8	64	0.00001	54000
目標値の事前計算	4×4	中規模	1.0	64	0.00001	282000
目標値の事前計算	2×2	中規模	0.8	64	0.00003	14000
目標値の事前計算	2×2	中規模	1.0	64	0.00003	8000
目標値の事前計算	4×4	小規模	0.8	32	0.0003	73000
目標値の事前計算	4×4	小規模	1.0	32	0.0003	55000

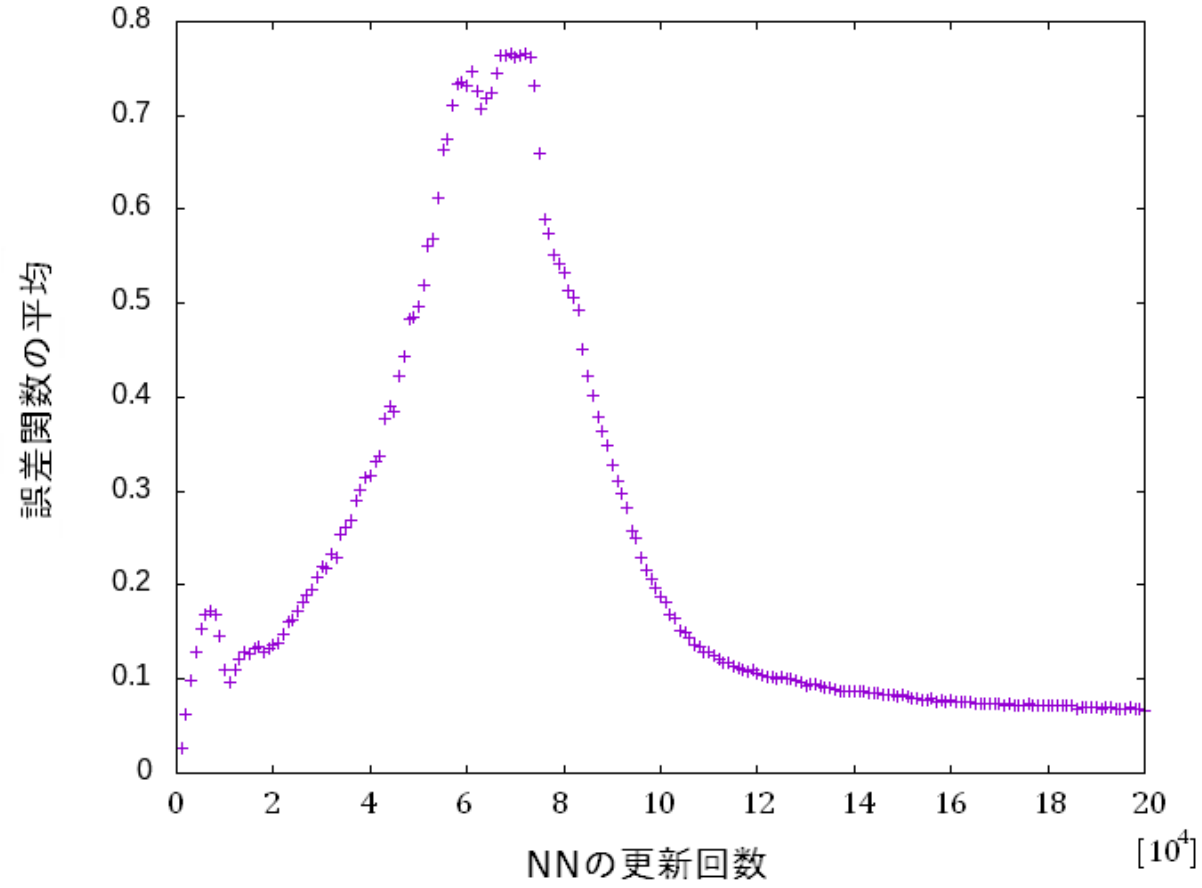

図 20: NN の誤差関数値の推移の例

イズは状態遷移 10^6 個，挙動方策の ε は 0.8，割引率は 0.99 とし，目標値の事前計算手法を用いた．なお，挙動方策による行動決定時と遷移後の状態における最善の行動決定時 (図 11 の 6 行目と 8 行目に相当) に所持していないアイテムを使おうとするなどの明らかに意味の無い行動は選ばないようにした．

それぞれ，1400000 回ネットワークの更新を行ったが，損失関数の値が時間経過で減少傾向になることは無く，挙動方策によるスコアの平均値が増加することもなかった．結果として，Rogue Clone III をプレイする人工知能の Q 学習は人工知能の性能を向上させることは一切なかった．学習がうまくいかなかった要因として著者が考えるものは以下の 3 点である．

・反復回数が足りない
・経験リプレイのメモリのサイズが小さい
・ResNet や中規模以上の NN が Q 学習に向いていない，またはハイパーパラメータの調節が困難

6　終わりに

本研究では大規模 NN を用いて既存人工知能の収益期待値の推定と Q 学習を試みた．既存人工知能の収益期待値の推定では良い精度 (決定係数 0.97 程度) を達成することができた．

一方でプレイヤの Q 学習においては良い結果は得られなかった．大規模 NN による Q 学習プログラムの実装には，小規模 NN での Q 学習における Q 学習アルゴリズムと，大規模 NN での収益推定における NN 構造と入力の関数をそのまま用いている．この 2 つの実験プログラムについては問題なく動作しているため，実装に大きな問題があるとは考えにくい．

小規模な NN に比べ大規模な NN は簡単なゲームの Q 学習においても成果が出なかったため，Q 学習を大規模 NN で行うこと自体が難易度が高いといえるのかもしれない．ただし，十分な数のミニバッチ学習を行えたとは言えないため，難易度が高いと言い切ることはできない．報酬の設計や Q 学習の発展手法においてもまだ考慮する点が多く残っていると考えられる．

謝辞

本論文を執筆するにあたり大変手厚く指導してくださった保木邦仁准教授に深く感謝致します.

参考文献

[1] Tesauro, G., Temporal difference learning and TD-Gammon, *Communications of the ACM*, Vol. 38, No. 3, pp. 58-68, 1995.

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Demis, H., Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7587, pp. 484-489, 2016.

[3] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D., Grandmaster level in StarCraft II using multi-agent reinforcement learning, *Nature*, Vol. 575, No. 7782, pp. 350-354, 2019.

[4] Brown, N., Sandholm, T., Superhuman AI for multiplayer poker, *Science*, Vol. 365, Issue 6456, pp. 885-890, 2019.

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, pp. 529-533, 2015.

[6] 岡谷貴之，“深層学習”，講談社，2015.

[7] Sutton, R. S., Barto, A. G., “Reinforcement Learning”, MIT Press, 2018.

[8] 牧野貴樹, 澁谷長史, 白川真一, 浅田稔, 麻生英樹, 荒井幸代, 飯間等, 伊藤真, 大倉和博, 黒江康明, 杉本徳和, 坪井祐太, 銅谷賢治, 前田新一, 松井藤五郎, 南泰浩, 宮崎和光, 目黒豊美, 森村哲郎, 森本淳, 保田俊行, 吉本潤一郎, これからの強化学習, 森北出版, 2016.

[9] Mauldin, M. L., Jacobson, G., Appel, A. W., Hamey, L. G. C., ROG-O-MATIC: a belligerent expert system, Carnegie Mellon University, CMU-CS-83-144, 1983.

[10] 金川裕司, 金子知適, ローグライクゲームによる強化学習ベンチマーク環境 Rogue-Gym の提案, Proceedings of The 23rd Game Programming Workshop, pp. 120-127, 2018.

[11] Kanagawa, Y., Kaneko, T., Rogue-Gym: A new challenge for generalization in reinforcement learning, Proceedings of the IEEE Conference on Games, 2019.

[12] 加納由希夫, 鶴岡慶雅, 内部報酬と Hybrid Reward Architecture を用いたローグライクゲームの強化学習, Proceedings of The 23rd Game Programming Workshop, pp. 64-71, 2018.

[13] 加納由希夫, 好奇心に基づく内部報酬を用いた強化学習によるローグライクゲームの学習, 修士論文, 東京大学大学院, 2020.

[14] 高橋一幸, ローグライクゲームにおける長期的戦略の学習, 修士論文, 北陸先端科学技術大学院大学, 2017.

[15] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, Vol. 362, Issue 6419, pp. 1140-1144, 2018.

[16] He, K., Zhang, X., Ren, S., Sun, J., Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Virsion and Pattern Recognition, pp. 770-778, 2016.

[17] 若月裕樹, Rogue Clone III をプレイする人工知能の開発, 卒業論文, 電気通信大学, 2019.

[18] Ioffe, S., Szegedy, C., Batch Normalization: Accelerating deep network training by reducing internal covariate shift, Proceedings of the 32nd International Conference on Machine Learning, PMLR 37: pp. 448-456, 2015.

[19] Kvalseth, T.O., Cautionary note about R^2, *The American Statistician*, Vol. 39, No. 4, pp. 279-285, 1985.

付録 A　Rogue Clone III の詳細ルール

Rogue Clone III で使用される用語やルールは非常に数が多い．この付録ではゲームのソースコードを読まないと分からないであろう詳細な設定まで紹介する．まず，ゲームをプレイする上で最も基本となるルールを紹介する．表 10 はプレイヤキャラクター (以下，プレイキャラ) の状態を表すステータスの項目である．ゲームの画面下に表示される情報とされない情報があり，メッセージを読むことでこれらの変化がわかるようになっている．表 11 はマップ，アイテム，モンスターなどの各オブジェクトの画面上での表記と説明である．表 12 はプレイヤが入力できるコマンドであり，キーボードで打ち込む文字とそのコマンドの内容を表している．

表 13 はモンスターのデータを表す．HP はダメージの許容量で，これを超えるダメージを受けるとモンスターは死亡する．経験値は死亡した際に，プレイキャラが手に入れる経験値の量である．アイテム落下率は死亡した際に，その場にアイテムを残す確率 (%) である．初期状態はプレイキャラが階層を移動した直後におけるモンスターの状態である．睡眠状態や熟睡状態は移動や攻撃をしないが，睡眠状態はプレイキャラが部屋を出入りした時や隣に来た時に 45%の確率で通常状態に戻る．擬態はアイテムに擬態しており，拾おうとすると正体を現し，通常状態になるというもの．攻撃能力と基礎命中率が関係する式は後述する．モンスターは 120 ターン経過するたびにプレイヤの見えない場所に発生するが，初期状態が必ず熟睡のモンスターは発生しない．また，表 14 はモンスターの特殊能力を示す．このうち炎は Rogue Clone III においてプレイキャラのいる方向へ正しく飛んでこないという不具合 (バグ) がある．

表 15 は食料の名称と効果を示す．slime-mold はオプション設定で名称を変更できる．

表 16 は武器の名称と性能を示す．命中性能と攻撃性能の説明は後述する．遠距離武器は複数個まとまって落ちており，投げて使い捨てにすることを前提としている．近接武器のように扱うこともできるが，arrow 以外は装備してから投げるとダメージと命中率が上がる．また，武器には補正値がついていることがあり，命中性能と攻撃性能の補正値の合計が -3 から $+3$ の間で落ちている．補正値がマイナスの物は必ず呪われている．呪われた武器，鎧，指輪は一度装備すると装備を外せなくなる．

表 17 は鎧の名称と性能を示す．鎧の Arm がプレイキャラの受けるダメージなどに関わる計算式は後述する．鎧にも補正値があり，-3 から $+3$ の補正がかかった状態で落ちていることがある．また敵の攻撃や罠の効果で錆びると補正値が 1 落ちる．武器と違い装備すれば補正値は判明する．

表 18 は指輪の名称と効果を示す．左右に同じ指輪を装備した場合，ring of teleportation

表 10: ステータス

用語	解説
level	現在の階層で 1F から 99F まである． 階段を上り下りすることで変化する．
Gold	拾った金塊の量でゲーム終了時のスコアになる． 死亡した場合 9 割に減少する．
Hp	プレイキャラの体力で現在 Hp と最大 Hp がある． 現在 Hp が 0 になると死亡しゲームが終了する．現在 Hp は一定の ターンが経過するごとに少しずつ回復する．
Str	敵に攻撃するときに参照される値で現在値と過去の最大値がある．
Arm	鎧の防御力で敵からの攻撃をどれだけ防げるかについて参照される．
Exp	プレイキャラのレベルと経験値で，一定の経験値を得ることで レベルが上がる．レベルが上がると HP が上がる． レベルの値は様々な計算に用いられる．
満腹度	隠しステータスで初期値は 1250 だが，1 ターンに 1 ずつ減っていき， 300 を切ると hungry，150 を切ると weak，20 を切ると faint と表示され， 20 以下ではプレイキャラが確率で動けなくなる．0 になると餓死する．
所持アイテム	プレイキャラが持ち運ぶアイテムで，最大 24 個まで所持できる． 遠距離武器は複数でも 1 個と数えられる．
盲目	状態変化でこの状態になると部屋の中や周り 1 マスが見えなくなり， 自分自身が歩いた場所しかマップに反映されない．
幻覚	状態変化でこの状態になるとアイテムとモンスターの表記が ランダムな文字になる．
混乱	状態変化でこの状態になると移動がランダムな方向になる．
浮遊	状態変化でこの状態になるとアイテムを拾えなくなり，階段を下れない． また，罠を踏まず，拘束されなくなる．
倍速	状態変化で 1 ターンに 2 回行動できるようになる．
鈍足	敵専用の状態変化で 2 ターンに 1 回しか行動ができなくなる．

表 11: オブジェクト

表示	解説
@	プレイキャラ
|-	部屋の壁で調べると隠し扉であることがある.
+	部屋の扉で調べないと見つからないものもある．プレイキャラが通行できる.
.	プレイキャラのいる部屋の何もない床だが見えない罠がある可能性がある. プレイキャラが通行できる.
(空白)	プレイキャラのいない部屋の床，もしくは何もない場所.
#	通路でプレイキャラが通行できる．調べないと見つからないものもある.
%	階段で上に乗ると階層を移動できるが，イェンダーの魔除けを持っていないと 下ることしかできない.
^	発見済みの罠．6 種類あり，踏むと一定確率で効果がある.
*	金塊で拾った量が Gold に加算される. このアイテムは所持アイテムの枠には入らない.
:	食料で食べると満腹度が回復する．2 種類ある.
!	水薬で飲むと何かの効果がある．14 種類ある.
?	巻物で読むと何かの効果がある．13 種類ある.
/	杖で振ると何かの効果がある．11 種類ある.
,	「Amulet of Yender」で 26 階以降に必ず落ちており，持っていると階段を登れる.
)	武器で装備したり投げることができる．装備すると攻撃能力に影響する. 8 種類あり，遠距離武器と近接武器がある.
]	鎧で装備できる．Arm の値に影響する.
=	指輪で左右の手に装備でき，何かの効果がある.
A-Z	モンスターで 26 種類いる．HP や獲得経験値，出現する階などが決まっている. また特殊な能力を持っている者がいる.

表 12: コマンド

入力	効果
.	何もせず時間経過する.
hjklyubn	周り 8 方向への移動，または攻撃をする.
m+方向	アイテムを拾わず移動する.
,	足元のアイテムを拾う.
Shift+方向	何かを踏むか壁に当たるまで連続移動する.
Ctrl+方向	何かに隣接するまで連続移動する.
s	隠されたものが周りにあるか調べる.
>	階段を下る.
<	階段を上る.
f+方向	自分が瀕死になるまで敵を攻撃する.
F+方向	どちらかが死亡するまで敵を攻撃する.
d+アイテム	アイテムを置く.
c+アイテム	アイテムに名前を付ける.
t+方向+アイテム	アイテムを投げる．敵に当たるとダメージを与える.
w+アイテム	武器を装備する.
W+アイテム	鎧を装備する.
T	鎧を脱ぐ.
P+アイテム	指輪を装備する.
R+左右	指輪を外す.
e+アイテム	食料を食べる.
r+アイテム	巻物を読む.
q+アイテム	水薬を飲む.
z+方向+アイテム	杖を振る.
^+方向	罠の種類を調べる.
i	所持アイテム一覧を表示する.
I+アイテム	指定したアイテムを表示する.
)	装備中の武器を表示する.
]	装備中の鎧を表示する.
?+コマンド	コマンドの確認する.
/+文字	文字の意味を確認する.
Ctrl+a	Hp の上昇平均を表示する.
Ctrl+p	メッセージの再送をする.
Esc	コマンドのキャンセルをする.
:	アイテムのコマンド入力前にアイテムの確認をする.
0-9	入力した数値の回数だけ次のコマンドを実行する.
o	オプション設定をする.
v	バージョンを見る.
S	ゲームを中断して，セーブする.
Q	ゲームを終了する.
!	シェルへ行く.

表 13: モンスター

表示	名称	出現階	HP	攻撃能力	基礎命中率	経験値	アイテム落下率	特殊能力	初期状態
K	kestral	1 – 6	10	1d4	60	2	0	飛行	通常/睡眠
E	emu	1 – 7	11	1d3	65	2	0		睡眠
B	bat	1 – 8	10	1d3	60	2	0	飛行/ふらふら	通常/熟睡
S	snake	1 – 9	8	1d3	50	2	0		通常/睡眠
H	hobgoblin	1 – 10	15	1d3/1d2	67	3	0		通常/睡眠
I	ice monster	2 – 11	15	0d0	68	5	0	凍結攻撃	熟睡
R	rattlesnake	3 – 12	19	2d5	70	10	0	毒攻撃	通常/睡眠
O	orc	4 – 13	25	1d6	70	5	10	金塊探し	通常/睡眠
Z	zombie	5 – 14	21	1d7	69	8	0		通常/睡眠
L	leprechaun	6 – 16	25	0d0	75	21	100	金塊盗み	熟睡
C	centaur	7 – 16	32	3d3/2d5	85	15	10		通常/熟睡
Q	quagga	8 – 17	30	3d5	78	20	20		通常/睡眠
A	aquator	9 – 18	25	0d0	100	20	0	水攻撃	通常/睡眠
N	nymph	10 – 19	25	0d0	75	39	100	アイテム盗み	熟睡
Y	yeti	11 – 20	35	3d6	80	50	20		通常/熟睡
F	venus fly-trap	12 – 99	73	5d5	80	91	0	拘束	不動
T	troll	13 – 22	75	4d6/1d4	75	125	33		通常/睡眠
W	wraith	14 – 23	45	2d8	75	55	0	レベル下げ攻撃	通常/熟睡
P	phantom	15 – 24	76	5d4	80	120	50	透明/ふらふら	通常/熟睡
X	xeroc	16 – 25	42	4d6	75	110	0		擬態
U	black unicorn	17 – 26	90	4d10	85	200	33		通常/睡眠
M	medusa	18 – 99	97	4d4/3d7	85	250	25	睨み	通常/睡眠
V	vampire	19 – 99	55	1d14/1d4	85	350	18	吸血攻撃	通常/睡眠
G	griffin	20 – 99	115	5d5/5d5	85	2000	10	飛行	通常/睡眠
J	jabberwock	21 – 99	132	3d10/4d5	100	3000	0		通常/熟睡
D	dragon	21 – 99	145	4d6/4d9	100	5000	90	炎	睡眠

表 14: モンスターの特殊能力

名称	効果
飛行	1 ターンに 2 回移動できる．1 回目の行動で移動している場合 2 回目に攻撃はできず，1 回目の行動で攻撃している場合 2 回目に移動はできない．
ふらふら	bat は 40%,phantom は 60%の確率で移動や攻撃がランダム移動になる．
凍結攻撃	攻撃時にプレイキャラを凍結させることがある．確率などは後述する．
毒攻撃	攻撃時にプレイキャラの現在 Str を 1 下げることがある． 確率などは後述する．
金塊探し	プレイキャラより金塊の方へ移動する． プレイキャラから攻撃を受けると解除される．
金塊盗み	攻撃時に金塊を盗み消える．落下アイテムが金塊になる．
水攻撃	攻撃時に鎧の補正値を 1 下げる． 鎧を脱ごうとするとプレイキャラより先に攻撃する．
アイテム盗み	装備中以外のアイテムを 1 つ盗み消える．
拘束	隣に来たプレイキャラを移動できなくさせる．攻撃のたびに 与えるダメージが 1 ずつ増えていく．自ら移動しない．
レベル下げ攻撃	攻撃時に 20%の確率でプレイキャラのレベルを 1 下げる． レベル 5 以下では発動しない．
透明	マップ上に表示されない． プレイキャラが攻撃されてもモンスターの名称が表示されない．
睨み	プレイキャラを視認できる時に 1 度だけ発動する．55%の確率で 成功し，プレイキャラが 12〜22 ターン混乱する．
吸血攻撃	40%の確率で攻撃時に最大 Hp と現在 Hp の両方，または Str の現在値 (50%の確率で最大値も)，またはその両方を 1 下げる．最大 Hp が 30 以下，または現在 Hp が 10 未満の時は 発動しない．また，Str の現在値が 3 以下の時は Str の現在値と最大値は下がらない．
炎	縦横斜めの直線上 7 マス以内の距離の位置にプレイキャラがいるとき， 50%の確率で炎を直線状に吐いて攻撃してくる．

表 15: 食料

名称	効果
ration	満腹度を 1/3 にした後 60%の確率で 950〜1150 回復し，40%の確率で 750〜950 回復した後，経験値 2 を加える．
slime-mold	満腹度を 1/3 にした後 950〜1150 回復する．

表 16: 武器

名称	命中性能	攻撃性能	注釈
short bow	1	1	
dart	1	1	遠距離武器
arrow	1	2	遠距離武器，short bow 装備時ダメージと命中率が上がる
dagger	1	3	遠距離武器
shuriken	1	4	遠距離武器
mace	2	3	
long sword	3	4	
two-handed sword	4	5	

表 17: 鎧

名称	Arm	注釈
leather armor	2	錆びない
ring mail	3	
scale mail	4	
chain mail	5	
banded mail	6	
splint mail	6	
plate mail	7	

表 18: 指輪

名称	効果
ring of stealth	眠っている敵を起こす確率が 1/(3+ 装備数) に減る.
ring of teleportation	毎ターン 8%の確率で別の場所へ移動する. 呪われている.
ring of regeneration	ターン経過による Hp の治癒量が 1 上がる.
ring of slow digestion	ターン経過時の満腹度が減少値が −1 される.
ring of add strength	補正値の分 Str の値が変動する. マイナスの場合, 呪われている.
ring of sustain strength	毒や吸血などで Str を下げられなくなる.
ring of dexterity	補正値 +1 に対して, 命中率が +2%, 回避率が +2%, 与えるダメージが +0.5, 罠の回避率が +1%, s コマンドの成功率が +1%, 凍結と毒の成功率を下げる. 補正値がマイナスの場合, 呪われている.
ring of adornment	1/2 の確率で呪われている. 呪われていない場合は nymph に盗まれるアイテムがこれになる.
ring of see invisible	phantom などの見えない敵が見えるようになる.
ring of maintain armor	鎧が錆びなくなる.
ring of searching	1 ターンに 2 回自動で周りを調べる.

表 19: 水薬

名称	効果
potion of increase strength	Str の現在値が 1 上がり， 現在値が最大値の場合最大値も上がる．
potion of restore strength	Str の現在値が最大値に戻る．
potion of healing	現在 Hp が回復する．現在 Hp が一定以上の時最大 Hp も上がる． 盲目状態を解除し，混乱や幻覚が治るまでの時間が半減する．
potion of extra healing	現在 Hp が回復する．現在 Hp が一定以上の時最大 HP も上がる． 盲目，混乱，幻覚の状態を解除する．
potion of poison	Str の現在値が 1〜3 下がる．幻覚を解除する．
potion of raise level	プレイキャラのレベルが 1 上がる． 経験値もそれに合わせ増加する．
potion of blindness	500〜800 ターンの間，盲目状態になる．
potion of hallucination	500〜800 ターンの間，幻覚状態になる．
potion of detect monster	現在いる階層の全てのモンスターが マップに表示されるようになる．
potion of detect things	現在いる階層の全てのアイテムがマップに表示される．
potion of confusion	12〜22 ターンの間，混乱状態になる．
potion of levitation	15〜30 ターンの間，浮遊状態になる．
potion of haste self	11〜21 ターンの間，倍速状態になる．
potion of see invisible	現在いる階層で，透明の敵が見えるようになる． 盲目状態を解除する．

表 20: 巻物

名称	効果
scroll of protect armor	装備中の鎧が錆びなくなり，呪いが解ける．
scroll of hold monster	プレイキャラの周り 2 マスのモンスターが熟睡する．
scroll of enchant weapon	装備中の武器の命中もしくは攻撃の補正値を 1 上げ，呪いが解ける．
scroll of enchant armor	装備中の鎧の補正値を 1 上げ，呪いが解ける．
scroll of identify	所持しているアイテムを 1 つ鑑定する．
scroll of teleportation	プレイキャラがランダムな場所に移動する．
scroll of sleep	2〜5 ターンの間，眠ってしまい行動できない．
scroll of scare monster	このアイテムが置かれている場所にはモンスターは攻撃または移動することができない．1 度置かれたこのアイテムを拾おうとすると消滅する．
scroll of remove curse	所持している全てのアイテムの呪いが解ける．
scroll of create monster	プレイキャラの隣にその階に出現するモンスターからランダムに選んだモンスターを作り出す．
scroll of aggravate monster	その階層のモンスターが全員起きた状態になる．
scroll of magic mapping	その階層の全ての地形がマップに表示される．
scroll of confuse monster	攻撃したモンスターを 1 度だけ 11〜21 ターン混乱させる状態になる．

表 21: 杖

名前	効果
wand of teleport away	振った先にいるモンスターをランダムな位置に移動させる.
wand of slow monster	振った先にいるモンスターを鈍足状態にする.
wand of invisible	振った先にいるモンスターを透明にする.
wand of polymorph	振った先にいるモンスターを別のモンスターに変える.
wand of haste monster	振った先にいるモンスターを倍速状態にする.
wand of magic missile	振った先にいるモンスターに攻撃と同じダメージを 与える. この攻撃は必ず命中する.
wand of cancellation	振った先にいるモンスターの特殊能力を消す.
wand of do nothing	何も起きない.
wand of drain life	通路上で使用すると振った先のモンスターに 現在 Hp の 1/3 のダメージを与え，現在 Hp が半減する. 部屋で使用すると部屋のすべてのモンスターに 現在 Hp の 1/3 のダメージを与え，現在 Hp が半減する.
wand of cold	振った方向へ氷が飛び出す．この氷は壁に当たるとランダムな 方向に跳ね返ることがある．敵に当たるとダメージを与え， 凍らせることがある. プレイキャラに当たるとダメージを受ける.
wand of fire	振った方向へ炎が飛び出す．この炎は壁に当たるとランダムな 方向に跳ね返ることがある．敵に当たるとダメージを与え， プレイキャラに当たるとダメージを受ける.

表 22: 初期アイテム

名称	注釈
ration of food	
ring mail	補正値 +1 装備中
mace	補正値命中 +1 攻撃 +1 装備中
short bow	補正値命中 +1
arrows	25〜35 のランダムな数量

表 23: 罠

名称	効果
trap door	踏むと 1 つ下の階層へ移動する.
bear trap	踏むと 4〜7 ターンの間移動ができなくなる.
teleport trap	踏むとランダムな位置に移動する.
poison dart trap	踏むと 1〜6 ダメージ受け，40%の確率で Str の現在値が下がる. Str の現在値が 3 未満では下がらない.
sleeping gas trap	踏むと 2〜5 ターン眠って行動できなくなる.
rust trap	踏むと鎧が錆びる.

を除き効果が重複する．指輪は共通で命中率が −1%，回避率が −1%，与えるダメージが −0.5 される．そのため実際は ring of dexterity などの効果は上記より低くなる．指輪は鑑定しない限り名称は判明せず，アイテム自体が ruby ring のような特徴でしか表記されない．一度判明すれば同じ指輪については名称が分かる．同じアイテムでも実際の名称と未判別時の特徴の対応はゲーム開始時にランダムに決定されているためゲーム毎に異なる．満腹度の減少値は本来指輪 1 つにつき，+0.5 であるが，Rogue Clone III ではバグにより +0.5 以上であれば必ず 1 ターンに満腹度が 2 減るようになり，ring of slow digestion の効果で −0.5 以下になると必ず満腹度が減らないようになるという不具合 (バグ) がある．

表 19 は水薬の名称と効果を示す．水薬は使用されるか鑑定されない限り，名称は判明せず，アイテム自体が red potion のような色でしか表記されない．一度判明すれば同じ水薬については名称が分かる．同じアイテムでも実際の名称と未判別時の色の対応はゲーム開始時にランダムに決定されているためゲーム毎に異なる．potion of healing と potion of extra healing の具体的な効果は後述する．

表 20 は巻物の名称と効果を示す． 巻物は使用されるか鑑定されない限り，上記の名称は判明せず，アイテム自体が scroll entitled: ‘bleach mep ogr’ のようなタイトルでしか表記されない．一度判明すれば同じ巻物については名称が分かる．未判別時の巻物のタイトルはゲーム開始時にランダムに決定されるためゲーム毎に異なる．別の効果の巻物でもタイトルが全く同一のものになる可能性がある．

表 21 は杖の名称と効果を示す．全ての杖には 3〜7 回の使用制限があり，それを超えて使用しても効果は発揮されない．杖は鑑定されない限り名称と使用回数は判明せず，アイテム自体が steel wand のような特徴でしか表記されない．一度判明すれば同じ杖については名称と使用回数が分かる．同じ杖でも実際の名称と未判別時の特徴の対応はゲーム開

表 24: Str の対応する値

Str の現在値 +ring of add strength の補正値	値
2–6	左上の項の式の値 −5
7–14	1
15–17	3
18	4
19–20	5
21	6
22–30	7
31–103	8

始時にランダムに決定されるためゲーム毎に異なる．

紹介したこれらアイテムは所持した時点で a から x の文字が割り当てられ，コマンドからアイテムを指定するときはこの文字を使用する．また，アイテムはクリア時に金塊に換金される．換金される額はアイテムによって異なるが，ここでは省略する．

表 22 はプレイキャラが所持するゲーム開始時の初期アイテムである．

表 23 は罠の名称と効果を示す．罠の個数と踏んだ時に罠を回避できる確率は後述する．罠を回避した時のみ罠の名称が判明しない．

以上のデータを補足する具体的な計算式を以下に記述する．

プレイキャラがモンスターに与える近接攻撃のダメージは以下の式で表される．(以下，特筆しない限り小数点以下切り捨てとする．)

$$\text{ダメージ} = \text{武器ダメージ} + \text{Str が対応する値} + (\text{プレイキャラのレベル} + \text{ring of dexterity の補正値} - \text{指輪装備数} + 1)/2$$

また，throw コマンドでモンスターに与えるダメージは以上の式の値を D として，装備していない物を投げた時は D，装備している dagger, dart, shuriken を投げた時は 3/2×D，arrow を装備中に bow を投げた時は arrow の D に bow の D を足し，2/3 倍した値となる．また，式中の武器ダメージは 1〜武器の攻撃性能値の内，ランダムな値を命中性能値の回数取得し足し合わせたものとなる．例として補正無しの long sword では 3〜12 の値となる．武器でない物は −1 となる．表 24 は Str の対応する値を示す．プレイキャラがモンスターに与えるダメージがマイナスの値の場合はモンスターの HP が増加する．

プレイキャラがモンスターに近接攻撃する時の命中率 (%) は以下の式で表される．

$$\text{命中率} = 40 + \text{武器の命中性能値} \times 3 + \text{プレイキャラのレベル} \times 2 + \text{ring of dexterity の補正値} \times 2 - \text{指輪装備数}$$

また，throw コマンドでの攻撃のモンスターへの命中率は以上の式の値を H として，装備していない物を投げた時は H，装備している dagger, dart, shuriken を投げた時と arrow を装備中に bow を投げた時は H の 4/3 倍の値となる．武器でない物を投げた時は武器の命中性能値は 1 として計算される．

モンスターがプレイキャラに近接攻撃する時のダメージは以下の式で表される．

$$\text{ダメージ} = \text{攻撃基本ダメージ} - \text{攻撃基本ダメージ} \times \text{Arm} \times 3 \div 100$$

攻撃基本ダメージは攻撃能力 mdn に対して，1〜n の内ランダムな値を m 回取得し足し合わせたものとなる．/で区切られている場合それらの合計値となる．例として centaur では 5〜19 となる．ただし階層が 52F 以降では以下の式に変化する．

$$\text{ダメージ} = \text{攻撃基本ダメージ} + \text{現在の階層} - 52$$

モンスターがプレイキャラに近接攻撃する時の命中率 (%) は以下の式で表される．

$$\text{命中率} = \text{基礎命中率} - \text{プレイキャラのレベル} \times 2 - \text{ring of dexterity の補正値} \times 2 + \text{指輪装備数}$$

ただし階層が 52F 以降では命中率は 100%となる．

凍結攻撃の成功率 (%) は以下の式で表される．

$$\text{成功率} = 88 \quad (\text{freeze\_percent} < 10 \text{ のとき})$$
$$\text{成功率} = 0 \quad (\text{上記以外})$$

この freeze_percent は以下の式で定義される．

$$\text{freeze\_percent} = 99 - \text{Str の現在値} \times 3/2 - (\text{プレイキャラのレベル} + \text{ring of dexterity の補正値}) \times 4 - \text{Arm} \times 5 - \text{最大 Hp} \div 3$$

凍結が成功すると 4〜8 ターン行動できず freeze_percent の確率でそのまま治ることなく死亡する．

表 25: 罠の数

現在の階層	罠の数
1,2	0
3–7	0–2
8–11	1–2
12-16	2–3
17–21	2–4
22–28	3–5
29–99	5–10

毒攻撃の成功率 (%) は以下の式で表される．

成功率 $=$ 0 (Str の現在値が 3 以下または, ring of sustain strength を装備している)

成功率 $=$ $71 - 6 \times \mathrm{Arm}$

罠を踏んだ際の罠の効果を受けずに済む回避率 (%) は以下の式で表される．

回避率 = プレイキャラのレベル + ring of dexterity の補正値

表 25 は階層に存在する罠の数を示す．

potion of heeling の具体的な効果は以下の通りである．

potion of heeling の効果

1. プレイキャラのレベルと同じだけ Hp が回復する．
2. この時点での現在 Hp/最大 Hp を ratio とする．

3-1. ratio=1.00 のとき，最大 Hp が 1 上がり，現在 Hp が全回復する．

3-2. 0.90≤ratio<1.00 のとき，現在 Hp が全回復する．

3-3. ratio<0.90 のとき，以下を実行する．

3-3-1. ratio<0.33 のとき，ratio=0.33 とする．

3-3-2. 現在 Hp に (最大 Hp − 現在 Hp) × ratio を加える．

4. 盲目状態ならば盲目を解除する．
5. 混乱状態ならば混乱が解けるまでのターン数を半減させる．(小数点以下切り上げ)
6. 幻覚状態ならば幻覚が解けるまでのターン数を半減させる．(小数点以下切り上げ)

表 26: Hp の回復までのターン数

プレイキャラのレベル	ターン数
1	20
2	18
3	17
4	14
5	13
6	10
7	9
8	8
9	7
10	4
11	3
12 以上	2

potion of extra heeling の効果は以下の通りである．

potion of extra heeling の効果

1. プレイキャラのレベルと同じだけ Hp が回復する．
2. この時点での現在 Hp/最大 Hp を ratio とする．

3-1. ratio=1.00 のとき，最大 Hp が 2 上がり，現在 Hp が全回復する．

3-2. 0.90≤ratio<1.00 のとき，最大 Hp が 1 上がり，現在 Hp が全回復する．

3-3. ratio<0.90 のとき，以下を実行する．

3-3-1. ratio<0.33 のとき，ratio=0.33 とする．

3-3-2. ratio を 2 倍する．

3-3-3. 現在 Hp に (最大 Hp − 現在 Hp) × ratio を加える．

3-3-4. 現在 Hp が最大 Hp を超えていたら最大 Hp まで戻す．

4. 盲目状態ならば盲目を解除する．
5. 混乱状態ならば混乱を解除する．
6. 幻覚状態ならば幻覚を解除する．

ターン経過によるHpの回復の頻度はプレイキャラのレベルによって異なり，一定のターンが経過するごとに現在Hpは1回復と2回復を交互に繰り返す．表26はHpが回復するまでのターン数を示す．

表27はプレイキャラのレベルアップに必要な経験値を示す．

階層に存在するモンスターの数は初期状態で4〜6匹であり部屋の中にランダムに配置される．階層に存在するアイテムの個数は金塊を除き，以下のアルゴリズムで求められる．

1/6の確率でn =2, 1/3の確率でn =3, 1/3の確率でn =4, 1/6の確率でn =5とし，33%の確率でnに1を加え続ける．最終的なnがアイテムの個数となる．

金塊以外のアイテムは部屋の中にランダムに配置される．アイテムごとに出現率が定められている．金塊は各部屋に46%の確率で1つ置かれている．後述する迷路には確実に金塊が落ちている．落ちている金塊の量は普通の部屋で現在の階層数の2倍〜16倍，迷路で現在の階層数の3倍〜24倍である．

通常の部屋とは異なるモンスターハウスという部屋が存在することがある．モンスターハウスが階層のどこかに存在する確率は一律で8%である．モンスターハウスには99%の確率で5〜10個のアイテムが階層にデフォルトで配置されたアイテムとは別に落ちている．また，99%の確率で(上記の確率とは別)アイテム数の2倍の数のモンスターが睡眠状態で配置される．アイテムが配置されなかった時は敵の数は22となる．ただし，部屋の面積以上にモンスターは配置されない．モンスターハウスに存在するモンスターは現在の階層を3で割った余りの数だけ先の階層のモンスターも出現する．モンスターハウスではプレイキャラが出入りした時のモンスターが起きる確率が75%と高い．また，モンスターハウスがある時に1%の確率で階層全体が大部屋となる．大部屋になると通路は存在しない．

部屋と通路に関するダンジョン生成のアルゴリズムを以下に紹介する．

表 27: レベルアップまでの累計必要経験値

プレイキャラのレベル	経験値
2	10
3	20
4	40
5	80
6	160
7	320
8	640
9	1300
10	2600
11	5200
12	10000
13	20000
14	40000
15	80000
16	160000
17	320000
18	1000000
19	3333333
20	6666666
21	10000001

ダンジョン生成のアルゴリズム

1. 大部屋になるときは，部屋の大きさを決め大部屋を設置する．
2. 大部屋にならないときは，まずフロアを以下のようにほぼ均等に 9 つの区間に分ける．

0	1	2
3	4	5
6	7	8

3. 手順 2. で分けられたそれぞれの区間についてランダムに範囲を狭める．
4. 確定で部屋になる区間を
   a. 0-1-2 b. 3-4-5 c. 6-7-8 d. 0-3-6 e. 1-4-7 f. 2-5-8
   の 6 パターンの中から 1 つ選ぶ．
5. 確定で部屋になる区間と，その他の区間をそれぞれ 60%の確率で部屋とする．部屋の範囲は手順 3. で決めた範囲と等しい．
6. 部屋にならなかった区間を一定の確率で迷路にする．迷路になる確率と迷路作成アルゴリズムは後述する．
7. すべての区間について，ランダムな順にその区間 i が部屋か迷路であるならば，区間 i+1，i+3，i+2，i+6 の順に別の区間に通路を繋げる．繋げようとした区間に部屋か迷路が無い，同じ行か同じ列の区間でない，または 1 つ飛ばしで繋げようとした時に間に部屋か迷路がある場合は失敗する．通路の生成ではまずドアを設置し，ドアとドアの間を途中で 2 回折れ曲がるように通路を引く．4%の確率で通路を重ねて繋ぐ．この時通路の一部が隠しマスになることがある．ドアは 12%の確率で隠される．(これらの隠しマスは s コマンドで発見できる)．全ての部屋が繋がればループを抜ける．
8. 部屋や迷路でない区間それぞれについて，ランダムな順に 50%の確率で通路を引く．引き方は後述する．

部屋にならなかった区間が迷路になる確率は以下の式で表される．

$$\text{確率} = 0 \quad (1\text{F})$$
$$\text{確率} = \text{現在の階層} \times 5/4 \quad (2\text{F}\sim14\text{F})$$
$$\text{確率} = \text{現在の階層} \times 9/4 \quad (15\text{F}\sim99\text{F})$$

迷路作成のアルゴリズムを以下に示す.

迷路作成のアルゴリズム

1. 迷路の区間内のある地点を開始場所にする.
2. その地点を通路 (#) とする.
3. 方角を順に上下左右と定める．20%の確率でこの方角の並び順をシャッフルする.
4. 手順 3. の方角の順に現在の位置から 1 マス移動した場所を考える．その場所が区間内であり，元の地点を除き，上下左右に通路が無い場所ならばその場所を新たな地点とし，手順 2.〜4. を再帰で実行する.
5. 区間内を 0,10,20 回のいずれかの回数，ランダムな位置を隠しマスに設定する.

部屋や迷路でない区間に通路を引くアルゴリズムを以下に示す.

部屋が無い場所から通路を引くアルゴリズム

1. 通路を引く区間を区間 i として区間 i−3，区間 i−1，区間 i+1，区間 i+3 をランダムな順に部屋か迷路があり，隣り合っているか調べる．部屋などがあり隣り合っていたら手順 2. に進む.
2. その区間の区間 i 方面の壁に通路が繋がっていなければ手順 3. に進む.
3. その区間の部屋か迷路に通路を繋げる．区間 i に通路 (#) が存在すればその中のランダムな位置から繋げ，無ければ区間の中心から繋げる.
4. 50%の確率で別の区間にも通路を引けたら引く．この時は必ず区間 i の中心から引く．もう 50%の確率で手順 5. へ進む.
5. 区間 i の上下左右をランダムな順に何もない区間か調べる．何もない区間が見つかればその区間へ区間 i の中心から通路を引く．その区間を新たな区間 i として手順 5. を繰り返す．周りに何もない区間が無ければ手順 6. へ進む.
6. 1 回でも手順 5. で通路を引いた場合，到達した区間から周りの部屋か迷路に対して手順 2.〜4. を実行する.

モンスターの移動パターンを以下に簡単に示す.

・モンスターは目的地を常に目指しながら移動する．目的地はプレイキャラか特定の場所である．モンスターの出現時点では目的地は部屋のいずれかのドアである.
・モンスターはまず目的地の方向へ移動しようとする．それができない場合，目的地方向が斜めの場合は後ろ 3 方向以外，目的地と同じ行，列の場合は前 3 方向と真後ろの中から

ランダムな順に移動を試みる．

・モンスターの目的地の変更されかたは以下の 4 点である．

1. プレイヤが視界に入った時，プレイヤを目的地にする．

2. 既に目的地の場合プレイヤを目的地にする．

3. 何もできずに 5 回以上場所を移動できないとき，プレイヤが目的地なら，ランダムな場所を目的地へ，特定の場所が目的地の場合はプレイヤを目的地とする．

付録B　AI-RC3 のゲーム内情報とインデックスの対応

AI-RC3 でインデックス (番号) を付けたアイテム，メッセージ，行動の対応表を示す.

表 28: アイテムのインデックス

番号	アイテム名	番号	アイテム名
0	gold	30	scroll of magic mapping
1	the amulet of Yender	31	scroll of confuse monster
2	ration of food	32	(unidentified scroll 1)
3	slime-mold	33	(unidentified scroll 2)
4	short bow	34	(unidentified scroll 3)
5	dart	35	(unidentified scroll 4)
6	arrow	36	(unidentified scroll 5)
7	dagger	37	(unidentified scroll 6)
8	shuriken	38	(unidentified scroll 7)
9	mace	39	(unidentified scroll 8)
10	long sword	40	(unidentified scroll 9)
11	two-handed sword	41	(unidentified scroll 10)
12	leather armor	42	(unidentified scroll 11)
13	ring mail	43	(unidentified scroll 12)
14	scale mail	44	(unidentified scroll 13)
15	chain mail	45	potion of increase strength
16	banded mail	46	potion of restore strength
17	splint mail	47	potion of healing
18	plate mail	48	potion of extra healing
19	scroll of protect armor	49	potion of poison
20	scroll of hold monster	50	potion of raise level
21	scroll of enchant weapon	51	potion of blindness
22	scroll of enchant armor	52	potion of hallucination
23	scroll of identify	53	potion of detect monster
24	scroll of teleportation	54	potion of detect things
25	scroll of sleep	55	potion of confusion
26	scroll of scare monster	56	potion of levitation
27	scroll of remove curse	57	potion of haste self
28	scroll of create monster	58	potion of see invisible
29	scroll of aggravate monster	59	(unidentified potion 1)

番号	アイテム名	番号	アイテム名
60	(unidentified potion 2)	90	(unidentified staff/wand 7)
61	(unidentified potion 3)	91	(unidentified staff/wand 8)
62	(unidentified potion 4)	92	(unidentified staff/wand 9)
63	(unidentified potion 5)	93	(unidentified staff/wand 10)
64	(unidentified potion 6)	94	(unidentified staff/wand 11)
65	(unidentified potion 7)	95	ring of stealth
66	(unidentified potion 8)	96	ring of teleportation
67	(unidentified potion 9)	97	ring of regeneration
68	(unidentified potion 10)	98	ring of slow digestion
69	(unidentified potion 11)	99	ring of add strength
70	(unidentified potion 12)	100	ring of sustain strength
71	(unidentified potion 13)	101	ring of dexterity
72	(unidentified potion 14)	102	ring of adornment
73	staff/wand of teleport away	103	ring of see invisible
74	staff/wand of slow monster	104	ring of maintain armor
75	staff/wand of invisible	105	ring of searching
76	staff/wand of polymorph	106	(unidentified ring 1)
77	staff/wand of haste monster	107	(unidentified ring 2)
78	staff/wand of magic missile	108	(unidentified ring 3)
79	staff/wand of cancellation	109	(unidentified ring 4)
80	staff/wand of do nothing	110	(unidentified ring 5)
81	staff/wand of drain life	111	(unidentified ring 6)
82	staff/wand of cold	112	(unidentified ring 7)
83	staff/wand of fire	113	(unidentified ring 8)
84	(unidentified staff/wand 1)	114	(unidentified ring 9)
85	(unidentified staff/wand 2)	115	(unidentified ring 10)
86	(unidentified staff/wand 3)	116	(unidentified ring 11)
87	(unidentified staff/wand 4)		
88	(unidentified staff/wand 5)		
89	(unidentified staff/wand 6)		

表 29: メッセージのインデックス

番号	メッセージ内容
1	宣言してゲームを終了しました
2	おいしい
3	まずい
4	現在 Str が 1 あがった
5	現在 Str が最大に戻った
6	回復した
7	大回復した
8	str が少し下がった
9	幻惑状態になった
10	盲目状態になった
11	階層にアイテム/モンスターが存在しなかった (potion of detect things/monster)
12	混乱した
13	浮き上がった
14	倍速になった
15	透明なモンスターが見えるようになった
16	scroll of scare monster を読んだ
17	周囲のモンスターが熟睡した
18	効果なし、読んだ巻物は hold monster だった
19	武器が強化された
20	効果なし、読んだ巻物は enchant weapon だった
21	鎧が強化された
22	効果なし、読んだ巻物は enchant armor だった
23	scroll of identify で何を鑑定するか
24	効果なし、読んだ巻物は create monster だった
25	巻物で眠った
26	鎧が錆びなくなった
27	効果なし、読んだ巻物は protct armor だった
28	全ての呪いが解けた
29	全てのモンスターが起きた
30	フロアの構造が分かった

番号	メッセージ内容
31	混乱攻撃可能になった
32	杖の不発
33	炎が出た
34	氷が出た
35	炎はモンスターに当たった (この後にモンスターの座標 (h, w) を送る)
36	氷はモンスターに当たった (この後にモンスターの座標 (h, w) を送る)
37	炎はモンスターに当たらなかった (この後にモンスターの座標 (h, w) を送る)
38	氷はモンスターに当たらなかった (この後にモンスターの座標 (h, w) を送る)
39	モンスターは凍り付いた
40	炎はプレイヤに当たった
41	氷はプレイヤに当たった
42	炎はプレイヤに当たらなかった
43	氷はプレイヤに当たらなかった
44	装備 (解除) 成功
45	呪いで装備解除に失敗
46	もう既に（指輪は 2 つ）装備してる
47	アミュレットは使えない (オリジナルには無いメッセージ)
48	指定のアイテムは無い
49	アイテムを置いた
50	既にアイテムが置かれている
51	アイテムを投げてモンスターに当てた
52	アイテムを投げてモンスターに当たらなかった
53	投げたアイテムは消滅した
54	アイテムを拾った
55	アイテムは xeroc だった
56	ここにアイテムはない
57	浮いててアイテムを拾えない
58	scroll of scare monster は消滅した
59	アイテムがいっぱいで拾えない
60	アイテムに乗った (この次にアイテムの情報を受け取る)

番号	メッセージ内容
61	攻撃を当てた
62	攻撃を外した
63	攻撃したモンスターは混乱した
64	ハエトリグサにつかまって動けない
65	罠にかかって動けない
66	罠は trap door
67	罠は bear trap
68	罠は teleport trap
69	罠は poison dart trap
70	罠は sleeping gas trap
71	罠は rust trap
72	そこに罠は無い
73	そこに階段は無い
74	浮いてて階段を下れない/アミュレットが無くて階段を登れない
75	モンスターを倒した
76	モンスターに攻撃された (この後にモンスターの種別を送る)
77	モンスターの攻撃を回避した (この後にモンスターの種別を送る)
78	faint で動けない
79	眠りや凍結、faint からの回復
80	幻惑が解けた
81	盲目が解けた
82	混乱が解けた
83	浮遊が解けた
84	倍速が解けた
85	罠は作動しなかった
86	下の階に落ちた
87	罠で移動できなくなった
88	矢が降ってきた
89	罠で眠らされた
90	水を浴びた

番号	メッセージ内容
91	鎧は錆びた
92	鎧は錆びなかった
93	プレイキャラは凍り付いた
94	金塊を盗まれた
95	アイテムを盗まれた
96	毒攻撃で現在 Str が下がった
97	吸血攻撃で最大 Hp が下がった
98	メデューサに混乱させられた
99	ドラゴンが炎を吐いた（この後に吐かれた地点の座標 (h, w) を送る）
100	レベルが上がった
101	レベルが下がった
102	クリアした
103	死亡した

表 30: 行動のインデックス

番号	行動内容
0	宣言してゲームを終了
1-8	8 方向への移動
9-16	アイテムを拾わずに 8 方向への移動
17	待機
18	サーチ
19	足元のアイテムを拾う
20	階段を下る
21	階段を上がる
22-45	指定のアイテムを置く
46-237	指定のアイテムを指定の方向へ投げる
238-429	指定のアイテムを (杖は指定の方向へ) 使う
430-437	指定の方向の罠の種類を調べる
438-461	指定のアイテムを鑑定する