TD学習を用いたテトリス解法アルゴリズム

NAKAYAMA, Ryoji / 中山, 亮士

(出版者 / Publisher)
法政大学大学院理工学・工学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 理工学・工学研究科編 / 法政大学大学院紀要. 理工学・工学研究科編
(巻 / Volume)
57
(開始ページ / Start Page)
1
(終了ページ / End Page)
8
(発行年 / Year)
2016-03-24
(URL)
https://doi.org/10.15002/00013283

法政大学大学院理工学・工学研究科紀要　Vol.57(2016 年 3 月）　　法政大学

TD 学習を用いたテトリス解法アルゴリズム

Tetris solution algorithm using the TD learning

中山　亮士
Ryoji NAKAYAMA
指導教員　平原誠

法政大学大学院理工学研究科応用情報工学専攻

A method using a genetic algorithm (GA) and neural networks (NN) was proposed in a previous study that focused on an algorithm for obtaining a Tetris solution. While this approach has a very high performance with respect to game score in Tetris, it has the drawback of taking a considerable amount of time for learning. The aim of this study is to reduce the learning time by online learning, using a TD learning method and NN.

Key Word : Tetris, GA, TD-Learning

1. はじめに

テトリスとは，2 次元平面上にブロックを効率よく配置していくゲームと捉えることができる．テトリスは最適化問題としてしばしば取り上げられており，様々な最適化手法が適用されてきた．この最適化問題は多様な場面での最適化問題に幅広く展開が可能である．トラックの荷物詰め込み作業の効率化等はその一例である．この分野での従来研究でよい結果を報告しているのが，非線形のニューラルネットワーク（以下 NN）と遺伝的アルゴリズム（以下 GA）を組み合わせた手法である[1]．しかし学習には約 500 万回ゲームの試行を行う必要がある．それゆえ，かなりの時間を費やさねばならないという欠点があった．

本研究の目的は，学習スピードを重視したテトリス解法アルゴリズムの提案である．学習スピードを短縮することにより，トラックの荷物詰め込みでの現場でも，イレギュラーな形の荷物が追加された場合でも即座に対応でき，現実的であると考えた．本研究では盤面を評価する関数に NN を使用し，NN の最適化に TD 学習を用いた新たな手法を提案する．

2. GA と NN を用いた従来研究

2.1　GA とは

GA とは，データを遺伝子で表現した「個体」を複数用意し，適応度の高い個体を優先的に選択して淘汰，交叉，突然変異の操作を繰り返しながら最適解を探索するアルゴリズムである．

2.1.1　淘汰

淘汰とは，現存している個体の中から次の世代に受け継がせる個体を選出するための操作である．この操作には，適応度の高い個体のみを選ぶエリート選択や，ある程度の確率で適応度の低い個体を選ぶルーレット選択等がある．

2.1.2　交叉

交叉とは，淘汰により選出された個体から，新たな個体を生成するための操作である．交叉方法としては，一様交叉法等がある．個体はデータ（遺伝子）の集合で表されるが，その各々のデータを 2 つの個体が一定確率で交換し合い，2 つの個体から新たな 2 つの個体を生成する手法である．以上のように新たな遺伝子の組み合わせを持つ個体を生成させる方法を交叉と呼ぶ．

2.1.3　突然変異

突然変異とは，交叉を行って個体を作成する際，その手順を無視してランダムな値のデータと置き換える手法である．これにより，同じデータのみで交叉を繰り返していく場合よりも，新たな遺伝子を持った個体が生まれないという状態（局初回）に陥りにくくなる．突然変異は一般的に小さな確率によって発生する．

2.2. GA と NN を用いた従来研究手法

従来研究として，GA と NN を組み合わせたものがある[1]．NN の入力には表 1 にある 8 個の特徴量を使用しており，出力はテトリスの盤面の評価値である．テトリミノの配置位置の決

定方法は，現在のテトリミノの回転と移動を考慮して仮配置し，NN の出力（評価値）を得て，

表 1　従来研究[1]で使用した 8 個の特徴量

名称	説明
Landing Height	直前に置いたピースの高さ
ErodedPieceCells	消えたラインの数×ピースの中で消えたブロックの数
RowTransitions	横方向にスキャンした時, セルの内容が変わる回数
ColTransitions	縦方向にスキャンした時, セルの内容が変わる回数
NumHole	穴の数
CumulativeWells	井戸の高さの階上の和
HoleDepth	穴の上のブロック数
RowWithHoles	穴のある行

最も高い評価値を持つ盤面に配置する．通常の NN は教師信号と出力との誤差を用いて学習するが，ここでは特殊な学習をしている．1 つの NN を 1 個体とし，GA によって NN の結合荷重を最適化する手法を取っている．個体の適応度は，個体によって表現される評価関数を用いて 100 回のテトリスゲームを行った時の平均消去段数としている．世代数 500，個体数 100 とし，淘汰，交叉，突然変異を繰り返し，最適化を行ったところ，結果として，約 6000 万消去段数の性能を示した．また，単純計算で 500 万回のテトリスゲームの試行を行わなければならず，学習時間が長いのが欠点といえる．

3. 提案手法①

GA と自作評価関数を用いた提案手法

4 つのサブ評価関数の線形結合で評価関数を構成し，各サブ評価関数にかかる係数（以下，評価係数）の値を GA で最適化する手法を提案した[2][3].

現在のフィールドの盤面から，落下中のテトリミノをどこに配置すれば良いかを探索する．

3.1 評価関数，評価係数

4 つのサブ評価関数$f_i(x)$(i=1,⋯,4)を設け，その線形和で盤面を評価し，最も良いとされる位置にテトリミノを置くものとする．評価$f(x)$は，

$$f(x) = \sum_{i=1}^{4} \alpha_i f_i(x)$$

と表すことができる．ここで$\alpha_i (i = 1,\cdots,4)$は評価係数である．後述の通り，$f_1(x)$と$f_2(x)$は大きくなるにつれフィールドの状況は悪化すると考えられるため，評価係数α_1，α_2は負とする．$f_3(x)$と$f_4(x)$は大きくなるにつれ，フィールドの状況は良くなると考えられるため，対応する評価係数α_3，α_4は正とする．

以下にサブ評価関数$f_i(x)$(i=1,⋯,4)の説明を示す．

$f_1(x)$：デッドスペースの数

デッドスペースとは，図 1 に示すように，上下がブロック，もしくはブロックと床に挟まれているスペースと定義する．この数が増加するほど，ゲームオーバーになる可能性が大きくなる．

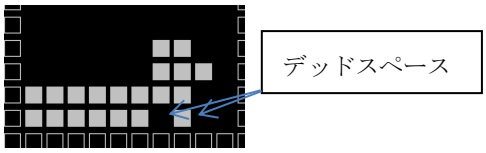

図 1.　デッドスペースの例

$f_2(x)$：　突出列の数

図 2 のようにフィールド全体の高さの平均から 4 段以上の差がある場合，その列を突出した高低差を持つ列とする．突出した高さを持つ列が多いと，安定したテトリミノ配置が困難になる．

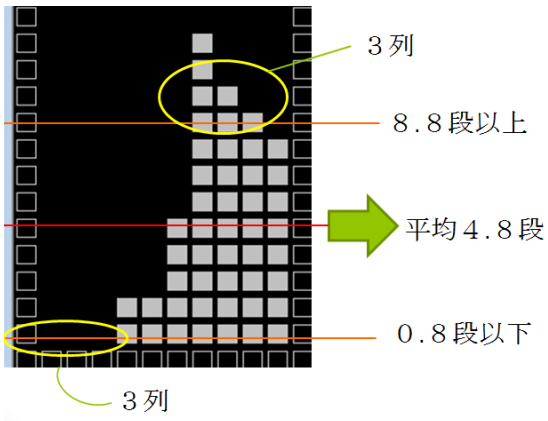

図 2.　突出した列を持つ盤面の例

$f_3(x)$：　高低差

各列において右隣の列との高低差の絶対値の合計を取る．その値を考慮した数値を返す．これはテトリミノ配置を考える際，凹凸が少なくなるようにテトリミノを置くための評価である．図 3 に合計高低差 4 の例を示す．

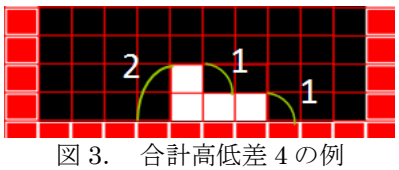

図 3.　合計高低差 4 の例

$f_4(x)$： 左右に 4 マスの溝を作成できる

図 4 のようにフィールド左右の壁際に縦 4 マス以上の隙間を作ることにより，複数列を同時に消すチャンスを作ることができる．

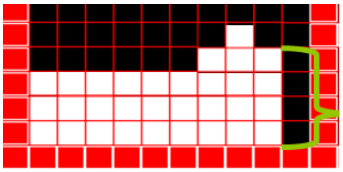

図 4．溝が作成された状態の例

3.2. 実験（GA と自作関数）

3.1 で記述した評価関数を用い，GA で最適化を試みる実験を行った．最適な評価係数α_iを個人の経験と憶測で決定するのは非常に困難である．このため，GA を用いて各評価係数の値を求めることとする．本研究では係数$\alpha_i(i = 1, \cdots,4)$を遺伝子とし，$\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$を 1 個体とし，これを 100 個体作成する．従来研究[1]と同様，各個体を用いてゲームを試行し，ゲームオーバーまでに消去することのできた段数をその個体の適応度とする．前述の通り，係数の値は正と負の 2 通りある．本実験ではα_1，α_2の値を－100 以上 0 以下，α_3，α_4の値を 0 以上 100 以下の整数とした．また，GA では個体数を 100 とし，100 世代実行した．その際も従来研究と同様，個体の適応度をより正確に測るため，1 個体につき 100 回の試行を行い，その平均を個体の適応度とした．

GA で評価係数の最適化を行った結果，最優秀個体の遺伝子(評価係数)は$\{-10, -95, 9, 16\}$となり，最高消去段数は 809,707 段であった．

一方，あらかじめ筆者が経験的に決定した評価係数の値は$\{-70, -30, 40, 10\}$である．これは筆者が，$f_1(x)$が最も重要であると考えたうえでの決定であった．この係数比でテトリスを 100 回試行したところ，最高消去段数は 6,385 段であった．

GA と筆者の係数の比較として，100 回試行した場合の最高消去段数と平均消去段数を表 2 に示す．

表 2.　100 回試行した消去段数の違い

	最高段数(段)	平均段数(段)
筆者	6,385	701.57
GA	809,707	162,837.40

表 2 より，筆者よりも GA の方が段を消去できていることが分かる．

また，図 5 は縦軸を消去段数，横軸を試行回数として，GA で求めた上記評価係数値を固定して 100 試行行った時の結果である．ばらつきがあるのが見て取れ，100 段以下の消去段数も 10 回ほどあった．アルゴリズムの問題なのか出現テトリミノの運の問題なのかは，図 5 からは判断できない．

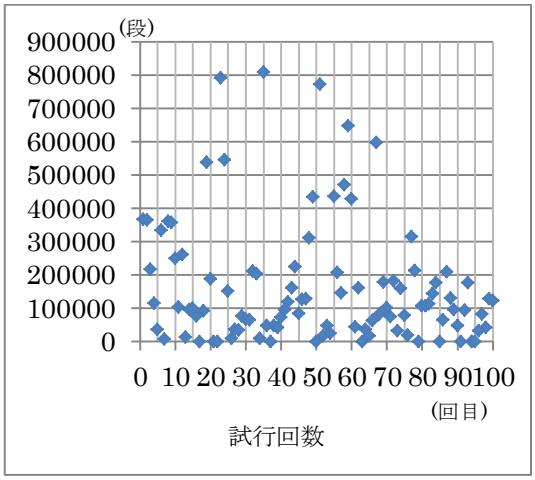

図 5 GA で求めた係数で 100 回試行した結果

3.3. 提案手法①まとめ，考察

本提案手法①では，4 つのサブ評価関数と，それに対応した評価係数との線形和を用い，盤面の評価を決定した．また，自動で評価係数の値を調整するために GA を用いた．このような枠組みは，複数の観点から評価し，その各評価の重みによって性能が変わるようなシステムを構築する時にも使用できるのではないかと考えられる．

消去段数の面で従来研究[1]に及ばなかったが，従来研究よりも 400 世代少ない 100 世代の学習である程度の消去段数を達成することができた．これは NN のパラメータ数が関係していると考えられる．従来研究では盤面の評価値を算出するのに NN の入力として 8 個の特徴量と中間層 50 個を使用しており，パラメータ数は $8 \times 50 + 50$の合計450であった．それに対し，提案手法①ではパラメータ数 4 で学習を行った．この計算量の差は非常に大きく学習時間に影響してくると予測できる．また，盤面のサブ評価関数を設計する際に，あらかじめ筆者が正か負かはっきりとした認識を持たせた結果であると考えられる．これにより，パラメータの取り得る値を正か負に限定できるため，解の探索空間を狭めることができる．消去段数の面で及ばない結果となったが，計算量に比べての消去段数については上々だったのではないかと考えられる．

今後，サブ評価関数をさらに増やし，より精密な評価設定をしていくことで，より効率的なテトリスの最適化アルゴリズムに近づくことが期待できる．しかし，係数値の決定において，

GA が最も適切だったとは言い切れない. 世代数が増えるにしたがって個体が優秀になるため, 1 試行においてのゲームオーバーまでの時間が長くかかり，90 世代以降になると 1 世代あたりの試行時間は約 12 時間となった. さらにサブ評価関数を増やし，世代数や個体数を増加すれば性能が良くなる可能性があるが，現実的に考えてトラックの荷物積み込み等の現場での使用は難しく，従来研究[1]と同じ課題が残った.

4. 強化学習[5]

強化学習には，方策，報酬関数，価値関数，そして環境のモデルという主に 4 つの構成要素がある.

方策とは, 学習のエージェントのふるまい方を決定する要素である. 本研究では環境モデルがテトリスとなるため，テトリミノをフィールド内のどこに配置するかを決定することを方策

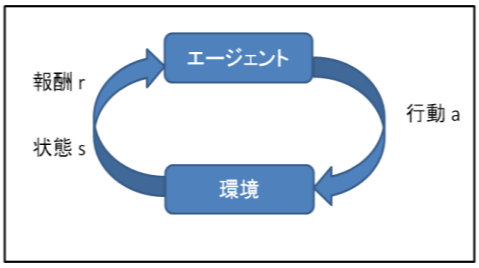

図 6　強化学習のイメージ

と呼ぶことができる.

報酬関数は強化学習において目標を決定する要素である. エージェントが方策に従って行動すると，正もしくは負の報酬が得られる. 一般的にエージェントは最終的に受け取る総報酬を最大化することを目的とする.

報酬関数が即時的な意味合いで何が良いのかを示すのに対し，価値関数は最終的に何が良いのかを指定する. すぐに得られる報酬だけでなく，長い目で見た場合の総報酬の量である.

図 6 のように，環境が状態s_tの場合，方策に従ってエージェントが行動aを起こし，それが環境に反映される. その環境の状態から，報酬関数により算出された報酬rをエージェント（もしくはエージェントがとった行動）に与えられる. そして状態s_{t+1}を受け取り, また方策に従って行動をとる. この報酬を最大にすることを目的として，エージェントは学習を行う.

4.1. TD(λ)

強化学習の中に TD 学習という手法が存在する. TD 法では最終結果を待たず，時刻$\mathrm{t}+1$で直ちに目標値を作り, 観測した報酬r_{t+1}と推定価値$V(s_{t+1})$を使用して適切な更新を行う. TD 学習は数ステップ先の価値と現在の価値の誤差を計算し，将来得られると予測される価値を現在の価値へ伝搬することができる.

TD 学習でよく用いられる TD(λ)では, TD 誤差と呼ばれる隣接する 2 つの状態間の推定価値の差及び報酬を用いて方策を調整し，学習を行う. TD 誤差は以下の式によって定義される.

$$\delta = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t).$$

γは割引率と呼ばれる係数$(0 \le \gamma \le 1)$であり，現在において将来の価値の重要度を決定する. この TD 誤差が正の時には，s_tの現在の状態価値$V_t(s_t)$よりも, s_tの真の状態価値$(r_{t+1} + \gamma V_t(s_{t+1}))$は高いということであり，負の時には$s_t$の状態価値の推定値よりも実際の状態価値は低いということになる. つまり，TD(λ)では TD 誤差を 0 にすることを目標として学習を行う. もっとも単純な TD 法は TD(0)と呼ばれ,

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t$$

で表される. αはステップサイズパラメータと呼ばれる学習率である$(0 \le \alpha \le 1)$.

また，NN で教師信号を$r_{t+1} + \gamma V_t(s_{t+1})$とし，出力を$V_t(s_t)$とした最急降下法を行うこともできる. $\overrightarrow{\theta_t}$を NN の結合荷重の列ベクトルとし，$\overrightarrow{\theta_t}$=$(\overrightarrow{\theta_t}(1), \overrightarrow{\theta_t}(2), \dots, \overrightarrow{\theta_t}(\mathrm{n}))^T$(T は転置を表す)で表現され，更新式は.

$$\begin{aligned}\overrightarrow{\theta_{t+1}} &= \overrightarrow{\theta_t} - \frac{1}{2}\alpha\nabla_{\overrightarrow{\theta_t}}[\mathrm{r}_{t+1} + \gamma \mathrm{V}_t(\mathrm{s}_{t+1}) - \mathrm{V}_t(\mathrm{s}_t)]^2 \\ &= \overrightarrow{\theta_t} + \alpha[\mathrm{r}_{t+1} + \gamma \mathrm{V}_t(\mathrm{s}_{t+1}) - \mathrm{V}_t(\mathrm{s}_t)]\nabla_{\overrightarrow{\theta_t}}\mathrm{V}_t(\mathrm{s}_t)\end{aligned}$$

と表される. $\nabla_{\overrightarrow{\theta_t}}\mathrm{V}_t(\mathrm{s}_t)$は，$f$を任意の関数とし，

$$\nabla_{\overrightarrow{\theta_t}}\mathrm{V}_t(\mathrm{s}_t) = \left(\frac{\partial f(\theta_t)}{\partial f(1)}, \frac{\partial f(\theta_t)}{\partial f(2)}, \dots, \frac{\partial f(\theta_t)}{\partial f(n)}\right)$$

と表す.

1 時刻先のみでなく，一定時刻先の学習結果を価値関数の推定値の更新として利用できるアルゴリズムは TD(λ)と呼ばれる. TD(λ)の大きな特徴として, TD 誤差による状態価値の修正を直前の状態だけでなくエピソード中に訪問したすべての状態に伝搬させることが挙げられる. それを可能としているのが適格度トレースであり，時刻 t における状態 s の適格度トレースは$\mathrm{e}_t(\mathrm{s})$と表される. 伝搬率はトレース減衰パラメータλによって制御されている$(0 \le \lambda \le 1)$. 適格度トレースは

$$e_t(s) = \begin{cases} \gamma\lambda e_t(s) & \mathrm{s} \neq s_t\text{の時} \\ \gamma\lambda e_t(s) + 1 & \mathrm{s} = s_t\text{の時} \end{cases}$$

と表される．状態s_tを訪問した時に$e_t(s)$の値が増加し，訪問しなかった場合，割引率γとλによって$e_t(s)$の値が減少することがわかる(ただし$\gamma < 1$もしくは$\lambda < 1$の場合)．これにより，状態 s をいつ，どのくらい訪問したかを記録することがでいる．この適格度トレースを用いた状態価値更新式は

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t(e_t)s_t$$

となる．λ=0 であれば t+1 時刻の即時報酬のみを得るために動き，λ=1 であれば，最終結果までの報酬を考えたアルゴリズムとなる．

4.2. TD-Gamon[6]

本研究の提案手法は，TD-Gamon と呼ばれるアプリケーションに使用されているアイデアを取り入れた．

TD-Gamon とは，バックギャモンと呼ばれるゲームのアプリケーションである．TD-Gamon は，予備知識をほとんど必要とせず、それでも世界最強のグランドマスターに近いレベルの手を指すことを学習する．この学習アルゴリズムは，TD(λ)と NN を直接的に組み合わせたものである．

バックギャモンは，世界中でプレイされている主要なゲームの 1 つである．このゲームは盤面上にポイントと呼ばれる 24 個の場所があり，その上に白黒それぞれ 15 個の駒を置いてゲームが行われる．図 7 はバックギャモンの典型的な初期配置の盤面である．2 人のプレイヤーがそれぞれ白と黒の駒（白駒，黒駒）を持ち駒とし，サイコロを振って任意の自分の色の駒を，出た目に合わせて移動させることができる．すべての駒をゴールまで導くことができれば勝利となり，すごろくの要素も備えた偶然性の高いゲームとなっている．

TD-Gamon は NN を使用している．NN の入力ニューロン数は 198 個である．バックギャモンの各ポイントに対して，そのポイントにある白駒の個数を 4 個のニューロンで示している．白駒が無ければ，4 個のニューロンすべてが値ゼロを取る．1 個（2 個，3 個）ある場合には，1 番目（2 番目，3 番目）までのニューロンが値 1 を取る．そのポイントに白駒が 4 個以上ある場合，3 番目までのニューロンの値 1，4 番目のニューロンは値(白駒数－3)/2 を取る．24 個それぞれのポイントにおいて，白駒に対して 4 個のニューロン，黒駒に対して 4 個のニューロンあり，合計 192 個のニューロンとなる．残りの 6 個のニューロンは，白の番か黒の番か，特殊な位置にある駒の個数等の付加的な情報を表している．このように盤面の状態を直接的に NN に取り込み，学習を行わせた．また，NN は 3 層で構成されており，中間層のニューロンは 40 個，出力層のニューロンは 1 個である．

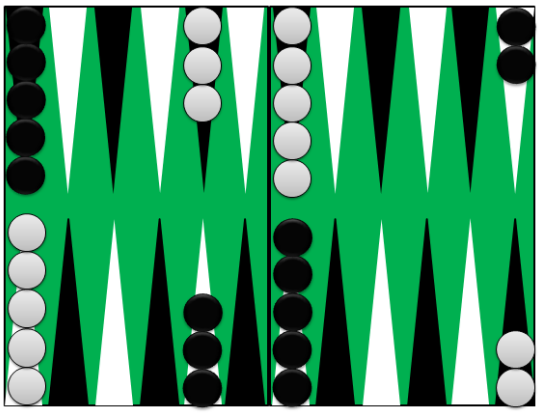

図 7　バックギャモンの初期配置

各入力ニューロンiからの信号x_iと対応する結合荷重w_{ji}との積が計算され，中間層ニューロン j で和が計算される．中間層ニューロン j の出力h_jは，重み付き和の非線形シグモイド関数である．

$$h_j = \frac{1}{1 + exp(-\sum_{i=1}^{198} w_{ji}x_i)}$$

中間層ニューロン j から出力ニューロンにかけての結合荷重c_jより,，出力ニューロンで和が計算される．出力層ニューロンの出力Vは

$$V = \frac{1}{1 + exp(-\sum_{j=1}^{40} c_j h_j)}$$

となる．TD 学習では，出力を$V_t(s_t)$とし，教師信号を$r_{t+1} + \gamma V_t(s_{t+1})$とした NN での最急降下法を行うことができる．結合荷重$w_{ji_{t+1}}$の更新式は．

$$w_{ji_{t+1}} = w_{ji_t} - \frac{1}{2}\alpha\nabla_{\overrightarrow{w_{ji_t}}}[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]^2$$
$$= w_{ji_t} + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]\nabla_{w_{ji_t}} V_t(s_t)$$

と表せる．適格度トレース$\overrightarrow{e_t}$は結合荷重w_{ji_t}の各要素と同じ次元のベクトルであり，

$$\overrightarrow{e_t} = \gamma\lambda\overrightarrow{e_{t-1}} + \nabla_{\overrightarrow{w_{ji_t}}} V_t(s_t)$$

で更新される．ただし，$\overrightarrow{e_0} = \vec{0}$である． TD(λ)の NN における更新式は

$$w_{ji_{t+1}} = w_{ji_t} + \alpha\delta_t\overrightarrow{e_t}$$

となる．結合荷重の値を更新し，結果的に状態価値関数$V_t(s_t)$を更新することができる．また，結合荷重w_{ji_t}の更新は中間層ニューロンjと出力ニューロンを結ぶc_jにも同様に適用され，適格度トレースもc_jと同じベクトルである．

バックギャモンのアプリケーションでは割引率$\gamma = 1$で，勝利を得た時点を除いて報酬は常に

0 であり，TD 誤差の部分は$V_t(s_{t+1}) - V_t(s_t)$となる．

TD-Gamon の学習のさせ方は，自らが白役と黒役を行う自己対戦形式であり，これにより膨大な回数のバックギャモンゲームの試行を容易に行えた．TD-Gamon は手を選ぶ際に，出たさいころの目に対して可能なプレイの仕方（大抵は 20 通り以上）をすべて仮配置し，NN でそれぞれの盤面に対する状態価値を推定し，最も高い状態価値を持つ盤面へ行く手を選択する．それぞれのゲームは，配置の系列s_0, s_1, s_2…を状態の系列とするような，1 つのエピソードとして扱われた．また，この TD 学習を，1 手毎に使用し，学習することとする．

ネットワークの初期荷重はランダムな小さい値に設定された．したがって，初期評価値は全くのランダムである．そのため，学習初期の対戦ではどちらも非常に弱く，一方が偶然に勝つような状況である．しかしゲームを繰り返すと，性能は急速に向上した．

自己対戦を 10 万回繰り返した後で，TD-Gamon は当時最強のバックギャモンのコンピュータプログラムと互角に戦えるほどとなった．

5 提案手法②

TD 学習と NN を用いた提案手法

従来研究[1]の GA と NN を用いた最適化は，学習に多くの時間を必要としていた．本提案手法では従来研究同様に NN を用いるが，学習時間を短縮するため，GA の代わりに，TD 学習を用いる．4.3 で記述したバックギャモンには勝ち負けの概念があった．しかし，テトリスはゲームオーバーになるまで無限に続くゲームである．そのため，勝利した時に報酬を与えるといった動作が適用されない．また，段を消去した時に報酬を与えてしまうと，段を消去することに貪欲になってしまい，客観的に盤面の状態が悪くなるにもかかわらず無理やり段を消去するような状態に陥ってしまう．よって，本提案手法では **NN** の出力を盤面のゲームオーバーになる確率とする．このため，以降状態価値を状態コストと呼び，状態コスト$V(s)$が低いほど良い盤面とする．また，報酬 r を罰金と呼ぶこととする．

図 8 のような，入力層ニューロン 209 個，中間層ニューロン 50 個，出力層ニューロン 1 個から構成される NN を用いた．入力をx_i(i=1,2,…,209)とし，入力層ニューロン i と中間層ニューロン j(j=1,2,…,50)をつなぐ結合荷重をw_{ji}とする．各入力ニューロンからの信号と対応する結合荷重との積が計算され，中間層ニューロンで和が計算される．中間層ニューロンの出力h_jは，非線形シグモイド関数であり，

$$h_j = \frac{1}{1 + exp(-\sum_{i=1}^{209} w_{ji} x_i)} \quad (1)$$

と表される．中間層ニューロン j から出力層にかけての結合荷重をc_jと表すと， NN の最終的な出力 V は，

$$V = \frac{1}{1 + exp(-\sum_{i=1}^{50} c_j h_j)} \quad (2)$$

となる．入力ニューロンのうち 200 個は，テトリスフィールド横 10 マス×縦 20 マスの合計 200 マス各々に対応している．もしニューロンに対応した 1 マスにブロックが入っていた場合，そのニューロンには 1 が入力され，ブロックが入っていなかった場合は 0 となる．さらに，表 3 に示した盤面の特徴量を取り込む入力ニューロンを 9 個用意し，盤面の付加的な情報を取得した．また，デッドスペースの定義は 3.1 で記した$f_1(x)$と同様である．

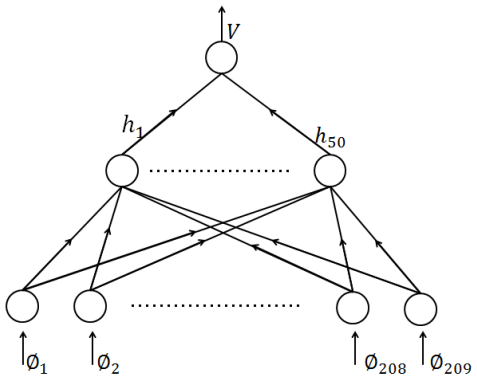

図 8．NN の構造

表 3．入力に用いた 9 個の盤面の特徴量

番号	特徴量
a	最大の高さ
b	ブロック数
c	デッドスペースの数
d	デッドスペースのある列の数
e	デッドスペースの上にあるブロック数
f	井戸の深さ
g	突出した列数
h	列のマスの変化
i	行のマスの変化

a)最大の高さ

ブロックが存在する一番高い行の値を入力とする．

b)ブロック数

フィールド内のブロックの合計を入力とする．

c)デッドスペースの数

デッドスペースの数の合計を入力とする．

d)デッドスペースのある列の数
　デッドスペースのある列の数の合計を入力とする．

e)デッドスペースの上にあるブロック数
　デッドスペースの上にあるブロックの数の総和を入力とする．

f)井戸の深さ
　ブロックとブロック(もしくはブロックと壁)に挟まれた幅 1 マスを井戸とし，フィールド内に存在する井戸の深さの合計を入力とする．

g)突出した列数
　3.1 で定義した$f_2(x)$と同様の定義である．突出した列の数を入力とする．

h)列のマスの変化数
　マスにブロックが入っていた場合を 1，入っていなかった場合を 0 とする．1 列ごとに列を参照し，フィールドの 20 行目から 0 行目にたどり着くまでにマスの値が変化した回数を求める．それを 10 列行い，その総和を入力とする．

i)行のマスの変化数
　マスにブロックが入っていた場合を 1，入っていなかった場合を 0 とする．1 行ごとに列を参照し，フィールドの左右の壁から壁までたどり着くまでにマスの値が変化した回数を求める．

　状態コスト$V_t(s_t)$の更新は，NN の結合荷重に TD 学習のアルゴリズムを用いることで可能としている．入力層ニューロンiから中間層ニューロン j までの結合荷重w_{ji_t}と，適格度トレースベクトル$\vec{e_t}$の更新式は以下のようになる．

$$w_{ji_{t+1}} = w_{ji_t} + \alpha\vec{e_t}\delta_t \tag{3}$$

$$\vec{e_t} = \gamma\lambda\vec{e_{t-1}} + \nabla_{w_{j_t}}V_t(s_t) \tag{4}$$

同様に，中間層ニューロン j から出力ニューロンにかけての結合荷重c_{j_t}の更新式は以下となる．

$$c_{j_{t+1}} = c_{j_t} + \alpha\vec{e_t}\delta_t \tag{5}$$

状態コストV(s)の算出方法は前述した式(1)と式(2)に従うものとする．
　罰金 r は，ゲームオーバー時にのみ 1 の値を取る．それ以外は 0 である．

$$r = \begin{cases} 1 & \text{ゲームオーバー時} \\ 0 & \text{それ以外} \end{cases} \tag{6}$$

これにより，ゲームオーバー時には最低評価の状態価値を受け取る罰金となる．

5.1 実験内容，実験結果

　本実験では，TD 学習により NN の結合荷重を最適化する実験を行った．事前実験より，それぞれの学習に関するパラメータは$\{\alpha, \gamma, \lambda\} = \{0.01, 1.0, 0.6\}$とした．
　縦軸を消去段数，横軸をゲームオーバーの回数とし，シード値 0,1,2 でそれぞれ 2000 回ゲームオーバーになるまで実行した結果を図 9～図 11 に示す．

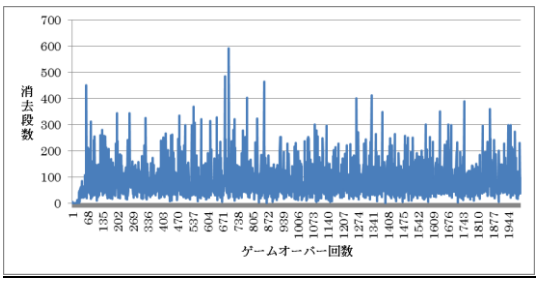

図 9　seed 値 0 の実行結果

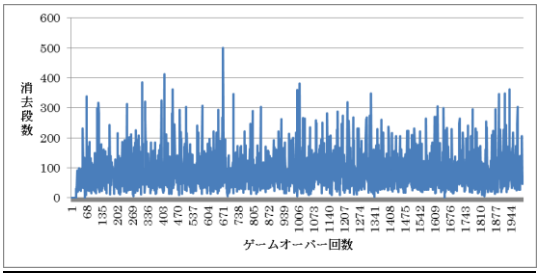

図 10　seed 値 1 の実行結果

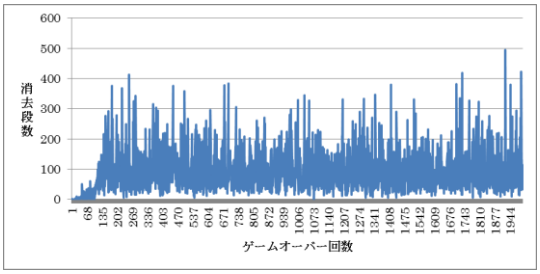

図 11　seed 値 2 の実行結果

　結果の図より，図 14，図 15 と図 16 から，実験開始直後は非常に性能が悪く，その後の数十回までは順調に消去段数が伸びていることがわかる．しかし，その後に伸びが無く，停滞していることがうかがえる．

5.2 提案手法②まとめ，考察

　TD を用いた提案手法では，段を消去し始めるまでのゲーム試行回数は非常に少なく，良好であった．しかしその後，消去段数に関して右肩上がりとなる結果を示すことができなかった．

原因として考えられるのは，テトリスの状態数の多さと強化学習の相性である．強化学習であるTD学習は，一度訪れた盤面の状態価値を伝搬させ，次に同じ状態に訪問する時に的確な報酬を与えていくものである．20×10マスといったテトリスのフィールドで一度訪れた盤面に再び出会うのは，数万回の施行を行ったとしても多くないのではと考えられる．さらに考えらえるのが，報酬の与え方である．一見バックギャモンとテトリスは盤面の状態価値(状態コスト)を決定しながら状態を推移していくといった類似したゲームと思われたが，TD-Gamonとの大きな違いとして，テトリスには勝ちという概念が無い．TD学習によりゲームオーバーにつながらない盤面にテトリミノを配置するような負の報酬(罰金)を与えた．これによりゲームオーバーにつながってしまう可能性のある悪い盤面を学習し，回避するように仕向けたのだが，真に良い盤面を学習するに至るまでにはすべての悪い盤面を経験し，誤差伝搬により伝搬しなくてはならない．よって，段消去時の正の報酬と，ゲームオーバー時の負の報酬を効率良く与えることのできる報酬付与方法を再考する必要があると考えられる．

また，TD学習の良い点として，学習時間が短時間で済むことがあげられている．また，GAでは学習時間が長いが，局所最適解に陥ることが少なく，性能の高い最適化を行うことができる．よって，この2つをうまく組み合わせるようなアルゴリズムを提案することによって，学習時間と性能の間を取った，マルチな提案手法が得られるのではないか．

参考文献

[1] 荒川正幹，宮崎真奈実(2012) ，ニューラルネットワークと遺伝的アルゴリズムを用いたテトリスコントローラの開発，情報処理学会 第74回全国大会講演論文，pp. 539-540.

[2] 中山亮士，平原誠(2015)，遺伝的アルゴリズムを用いたテトリスの解法アルゴリズム，電子情報通信学会2015年総合大会，ISS-P-51.

[3] 中山亮士，平原誠(2015)，遺伝的アルゴリズムのテトリスへの適用，第67回知的システム研究会.

[4] 中山亮士，平原誠(2016)，TD学習を用いたテトリスの解法，電子情報通信学会2016年総合大会(予定).

[5] R.S.Sutton and A.G.Barto(1998.), Reinforcement Learning, MIT Press.

[6] Gerald Tesauro(1995) ,Temporal Difference Learning and TD-Gammon ,Communication of the ACM,vol.38,pp.58-67.