北里大学

理学部物理学科学士論文

Social learning tournament と最適戦略のアルゴリズム

平成 26 年　月　日

非線形物理学講座

学籍番号　SP－10104

大越　創真

１　研究目的

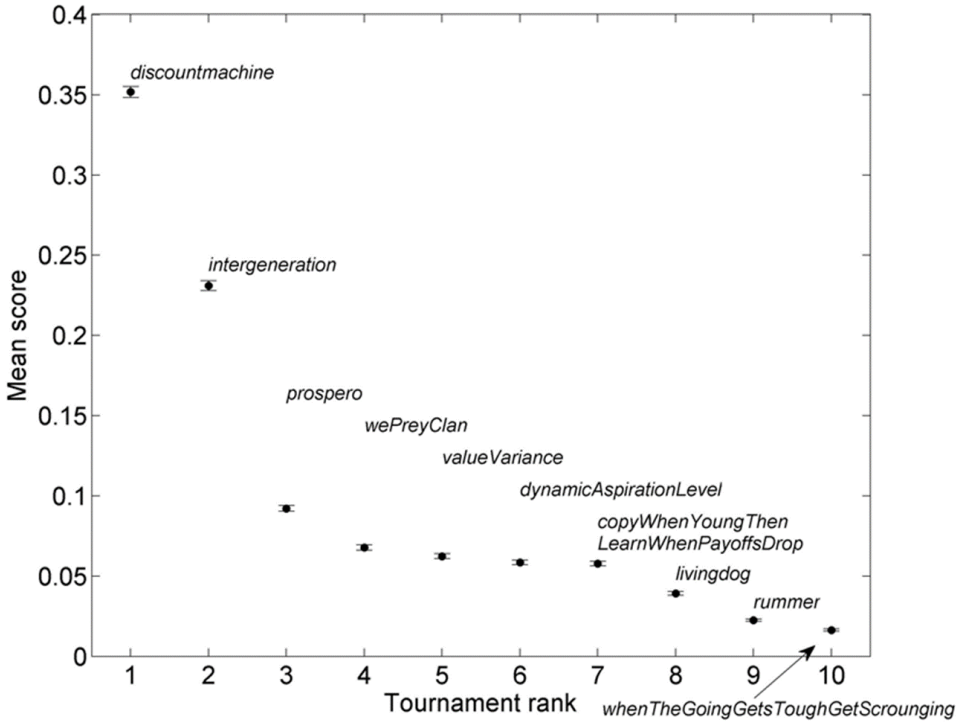
「最も効率の良い学習方法は social learning、いわゆるカンニングである。」
このことに関してすでに多くの論文が発表されている。しかし、やみくもに他のコピーばかりでもいい結果が得られないことも研究によってわかっている。では実際にどのように行動するのが一番いいのか。複雑に時間変化する環境で最適な振る舞いを競い合うコンピュータプログラムのトーナメントを開催し、優秀なプログラムのアルゴリズムを解析した L.Rendell らの論文がある。そのトーナメントは、(1),トライ＆エラーで最適な振る舞いを探す（innovate）こと、(2),他のエージェントの行動を知ること(observe)、(3),(1)と(2)で得た情報を用いて選択しリターンを得る(exploit)、の 3 つの行動を組み合わせるプログラム同士で総リターンを競い合うものであった。本発表では、このトーナメントの勝者であった discount マシンと呼ばれるプログラムのアルゴリズムに焦点を当て、最適の情報のコピーの方法について解説する。

２　トーナメントのルール

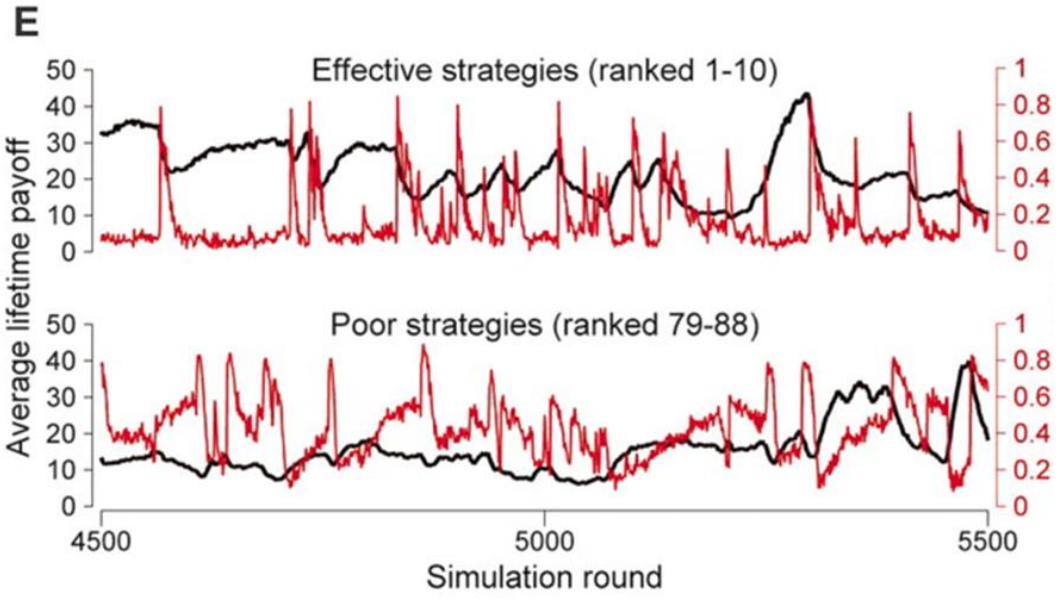
世界中から募集した百数十もの戦略を、無数のレバーを持つスロットマシンを用いたコンピュータシミュレーションのトーナメントで戦わせ、その成績を競うものである。
実際に戦略がとることのできる行動は、出る枚数のわかっているレバーを引いてコインを獲得する exploit、１つの知らないレバーの情報を調べる innovate、他人の引いた複数のレバーの情報を調べる observe の３つである。また、レバーの情報は一定の確率 pc によって変化する。つまり、レバーの情報は１ターン進むごとに変わってしまう可能性、いわゆる不確定性がある。そして決められたターン数の中でより多くのコインを獲得していく。ただし、各戦略は 1/50 の確率で他戦略または自戦略に生まれ変わってしまう。この時、自戦略に生まれ変わることのできる確率は、その戦略が生まれ変わるまでに獲得したコインの枚数に比例して高くなる。つまり、より多くのコインを獲得できる戦略は、より長く自戦略でいられる可能性が高いということである。このトーナメントでは、複数の戦略を１つの環境の中で決められたラウンド数行動させて、トーナメント終了時の各戦略の個体数の割合、いわゆる占有率が高い戦略の勝利となる。
トーナメントは、stage1 と stage2 に分かれていて、stage1 では各戦略の初期個体数 50vs50 の総当たり戦を行い、その上位 10 チームが stage2 に進む。stage2 では、上位 10 チームを同じ環境で戦わせ占有率を競った。

3　トーナメントの結果

トーナメントの stage2 の最終結果は以下の図になった。



また以下の図は、このトーナメントの stage1 において上位 10 チームと 79~88 位のチームの 10 ラウンドごとの学習(observe または innovate)の割合を赤い線で、その時点での平均獲得コイン枚数を黒い線で示したものである。



４　トーナメント結果の考察

トーナメントの stage1 において上位 10 チームと 79~88 位のチームの 10 ラウンドごとの observe または innovate の割合を赤い線で、その時点での平均獲得コイン枚数を黒い線で示した図の上下のグラフを比べてみる。
上位 10 チームのグラフでは平均獲得コイン枚数を表す黒い線が大きく下がったところで逆に学習の割合を表す赤い線が大きく上がっている。つまり、exploit によって得られるコイン数が下がったときに、集中して学習を行っていて、そのほかのときはほとんど学習していないことがわかる。しかし、79~88 位のチームでは下がっている場合だけでなく、全体的に学習を行っていることがわかる。

つまり、上位のチームは自分の持っている情報がいい情報であるかどうかはっきり見極めることができているということがいえる。

では、持っている情報の善し悪しはどのようにして判断しているのだろうか。

このトーナメントにおいて最も優秀な成績を収めた discount マシンがどのようなプログラムコードで動いているのか解析し、そのアルゴリズムについて考察する。

５　discount マシンのアルゴリズム解析

Discount マシンはトーナメントによって管理されている戦略ごとの行動・情報の履歴を用いて行動を決定するために必要な数値計算を行っている。

ラウンド数、今までの行動とその結果をラウンドごとに下記のように mh に記録する。また新しいレバー情報を手に入れたとき、下記のように mr に記録する。

$$mh = \begin{pmatrix} ラウンド数 \\ 行動 \\ レバー番号 \\ レバー報酬 \end{pmatrix}, \quad mr = \begin{pmatrix} レバー番号 \\ レバー報酬 \end{pmatrix}$$

mh の行動には exploit のとき 1、observe のとき 0、innovate のとき-1 と記録する。

Discount マシンのアルゴリズム

discount マシンの個体が生まれてから経過したラウンド数を T とする。
T = 0 のとき、つまり生まれ変わってすぐのときは observe を行う。
T = 1 のときで mr に情報がない場合は innovate を行う。

それ以外は以下のように行動する。

---

mh の 1 行目が 1 の列数を数える。このアルゴリズムでは、最初のラウンドに必ず observe しているので、列数が observe した場合に得られる情報の最大の数となる。
今までに得た最大の報酬として、mh の 4 行目より最も大きい値を Mp に代入する。

Discount マシンのアルゴリズムは、各レバーに対して exploit した場合の獲得コインの期待値 bestpayoff を計算し最も効率の良いレバーを決定し、exploit、observe それぞれの行動を行った場合の獲得コインの期待値を計算して最終的な行動を決定している。その計算に必要な突然変異の確率 pc、レバーの平均コイン数 mean についてまず求める。

＜突然変異の確率 pc を推定する関数＞

mh より mr に記録されているレバー番号ごとに情報を抜き出し、またその中で observe しているときの情報は除外する。この時のレバー番号を n とする。抜き出した情報の 4 行目のレバー報酬が変化した回数を Cn、変化しなかった回数を NCn に代入する。ただし、NCn > 12 の場合は NCn = 12 とする。また変化した直後の値を Mn に代入する。突然変異の確率 pc はレバーを引いた回数とレバーの変化した回数より求めることができるため、以下の式で推定できる。またレバーの情報平均コイン数 mean についてラウンドごとに以下の式で推定する。

$$pc = \frac{各レバーの\boldsymbol{Cn}の合計}{各レバーの\boldsymbol{Cn}と\boldsymbol{NCn}の合計}$$

$$mean = \frac{各レバーの初期値と\boldsymbol{Mn}の合計}{各レバーの(\boldsymbol{Cn}+1)の合計}$$

ただし mean について、T < 6 のとき、データ数が少ないので以下の式で推定する。

$$mean = \frac{レバー報酬の合計}{T}$$

[突然変異の確率 pc を推定する関数の例]

mh、mr が以下の値を持つ場合の関数の動きを見る。

mr

$$\begin{pmatrix} 12 & 6 & 9 & 24 & 19 \\ 4 & 2 & 3 & 5 & 7 \end{pmatrix}$$

mh

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & -1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 12 & 6 & 9 & 9 & 24 & 9 & 9 & 9 & 9 & 19 \\ 4 & 2 & 11 & 11 & 5 & 11 & 11 & 11 & 3 & 7 \end{pmatrix}$$

まず、mh の情報をレバー番号ごとに抜き出す。

$$\begin{pmatrix} 1 \\ 0 \\ 12 \\ 4 \end{pmatrix}\begin{pmatrix} 2 \\ -1 \\ 6 \\ 2 \end{pmatrix}\begin{pmatrix} 3 & 4 & 6 & 7 & 8 & 9 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 9 & 9 & 9 & 9 & 9 & 9 \\ 11 & 11 & 11 & 11 & 11 & 3 \end{pmatrix}\begin{pmatrix} 5 \\ 0 \\ 24 \\ 5 \end{pmatrix}\begin{pmatrix} 10 \\ 0 \\ 19 \\ 7 \end{pmatrix}$$

2 行目の値が 0(observe)の情報は除外する。
このときのレバー報酬が前回と変わっている場合は Cn に+1 して、変わっていない場合は NCn に+1 する。
また赤字で示した各レバーの初期値と情報が変化した直後の値の合計をそれぞれレバー番号ごとに Mn に代入する。

$$\begin{pmatrix} 2 \\ -1 \\ 6 \\ 2 \end{pmatrix}\begin{pmatrix} 4 & 6 & 7 & 8 & 9 \\ 1 & 1 & 1 & 1 & 1 \\ 9 & 9 & 9 & 9 & 9 \\ 11 & 11 & 11 & 11 & 3 \end{pmatrix}$$

NC9　　C9

以上より、Pc = $\frac{C9}{NC9+C9} = \frac{1}{4}$　　mean = $\frac{M6+M9}{(C6+1)+(C9+1)} = \frac{2+14}{(0+1)+(1+1)} = 5.333\ldots$
となる。

ここで、行動決定に必要な observe で得られる獲得コイン数の情報の平均を計算する。

Mh の 2 行目が 0(observe)のときの情報を抜き出す。
抜き出した情報の 4 行目のレバー情報の平均を Om に代入する。もし、Om の値が 0 なら Om に mean を代入する。

次に求めた pc より bestpayoff を計算し、最も効率の良いレバーを推定する。

＜最も効率の良いレバーを推定する関数＞

mh より、mr に記録されているレバー番号ごとに情報を抜き出す。この時、レバーの最後に行った行動によって期待値の計算が異なる。
① 抜き出した情報の 2 行目の最後の値が 0(observe)のとき。
情報の列数が 10 より多い場合、抜き出した情報を最も新しいものから 10 番目までの情報だけにする。その中で、最後に observe 以外の行動をしたとき以降のデータだけを残す。この時、レバーが最後に使われたラウンド数を Rn に代入する。抜き出した情報の列数を m 列として、1~m 列に対して抜き出した情報の 4 行目のレバー報酬を RBm に、1 行目のラウンド数を Rm にそれぞれ代入する。各列のレバー報酬の期待値 Pm は以上の値より以下の計算で推定する。

noise(m) = (1-pc)**(Rn-Rm)
Pm = RBm*noise(m)

noize(m)はその情報がレバーの中で最も新しい情報に比べどのくらいの信憑性があるのかについて計算している。
以上よりレバー番号を n としてレバー報酬の期待値 BPn は以下の式で推定できる。

$$BPn = \frac{Pm\text{の合計}}{noize(m)\text{の合計}}$$

② 抜き出した情報の 2 行目の最後の値が 0(observe)以外のとき。
レバー番号を n として抜き出した情報の 4 行目の最後の値を BPn に代入する。

以上で求めたレバー報酬の期待値 BPn より、bestpayoff を以下の式で推定する。

bestpayoff(n) = (1-(1-pc)**(T-R))*mean+(((1-pc)**(T-R))*BPn)

bestpayoff(n)のなかで値が最大のものを bestpayoff に代入し、その時の n を bestmove に代入する。

[最も効率の良いレバーを推定する関数の例]

mh、mr が以下の値を持つ場合の関数の動きを見る。

mr

$$\begin{pmatrix} 12 & 6 & 9 & 24 \\ 4 & 2 & 3 & 8 \end{pmatrix}$$

mh

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 12 & 6 & 9 & 9 & 24 & 9 & 11 & 9 & 9 & 24 \\ 4 & 2 & 11 & 11 & 8 & 3 & 5 & 4 & 7 & 8 \end{pmatrix}$$

まず、mh の情報をレバー番号ごとに抜き出す。

$$\begin{pmatrix} 1 \\ 0 \\ 12 \\ 4 \end{pmatrix}\begin{pmatrix} 2 \\ -1 \\ 6 \\ 2 \end{pmatrix}\begin{pmatrix} 3 & 4 & 6 & 8 & 9 \\ 0 & 1 & 1 & 0 & 0 \\ 9 & 9 & 9 & 9 & 9 \\ 11 & 11 & 3 & 4 & 7 \end{pmatrix}\begin{pmatrix} 5 & 10 \\ 0 & 1 \\ 24 & 24 \\ 8 & 8 \end{pmatrix}$$

① 抜き出した情報の 2 行目の最後の値が 0(observe)のとき。

抜き出した情報の 2 行目が 0(observe)以外の列以降だけ残す。

$$\begin{pmatrix} 1 \\ 0 \\ 12 \\ 4 \end{pmatrix}\begin{pmatrix} 8 & 9 \\ 0 & 0 \\ 9 & 9 \\ 4 & 7 \end{pmatrix}$$

レバー番号12の情報は列数が1なので、1行目の値をR1、4行目の値をRB1としてnoize(1)、P1、BP12、を計算する。

noise(1) = (1-pc)**(R-R1) = 1

P1 = RB1*noise(1) = 4

$$\text{BP12} = \frac{\text{P1}}{\text{noize(1)}} = 4$$

レバー番号 9 の情報は列数が 2 なので、1 行目 1 列目の値を R1、1 行目 2 列目の値を R2、4 行目 1 列目の値を RB1、4 行目 2 列目の値を RB2 として noize(1)、noize(2)、BP1、BP2 を計算する。

noise(1) = (1-pc)**(R-R1) =(1-0.25)**(9-8) = 0.75　noize(2) = (1-pc)**(R-R2) = 1

P1 = RB1*noize(1) = 4*0.75 = 3　p2 = RB2*noise(2) = 7

$$\text{BP9} = \frac{\text{P1+P2}}{\text{noize(1)+noize(2)}} = \frac{3+7}{1+0.75} = 5.714\ldots$$

② 抜き出した情報の 2 行目の最後の値が 0(observe)以外のとき。

$$\begin{pmatrix} 2 \\ -1 \\ 6 \\ 2 \end{pmatrix}\begin{pmatrix} 5 & 10 \\ 0 & 1 \\ 24 & 24 \\ 8 & 8 \end{pmatrix}$$

各レバーの最後のレバー報酬をそれぞれ BP6、BP24 に代入する。

以上で求めた R6、BP6、R9、BP9、R12、BP12、R24、BP24 を用いてそれぞれのレバーの獲得コインの期待値 bestpayoff(n)を計算する

bestpayoff(6) = (1-(1-pc)**(T-R6))*mean+(((1-pc)**(T-R6))*BP6)
= (1-(1-0.25)**(11-2))*5.333+(((1-0.25)**(11-2))*2)
=
bestpayoff(9) = (1-(1-pc)**(T-R9))*mean+(((1-pc)**(T-R9))*BP9)
= (1-(1-0.25)**(11-9))*5.333+(((1-0.25)**(11-9))*5.714)
=
bestpayoff(12) = (1-(1-pc)**(T-R12))*mean+(((1-pc)**(T-R12))*BP12)
= (1-(1-0.25)**(11-1))*5.333+(((1-0.25)**(11-1))*4)
=
bestpayoff(24) = (1-(1-pc)**(T-R24))*mean+(((1-pc)**(T-R24))*BP24)
= (1-(1-0.25)**(11-10))*5.333+(((1-0.25)**(11-10))*8)
=

求めた各レバーの獲得コインの期待値を比べ、最も大きな値を bestpayoff に代入し、そのレバー番号 n を bestmove に代入する。

これまでに計算した値より、bestmove に代入されているレバー番号のレバーを exploit した場合の獲得コイン数の期待値と、observe した場合の獲得コイン数の期待値を計算する。

その計算をするために 1 ラウンドごとにレバー情報が保たれる確率 keep について定義する。keep は突然変異の確率 pc と、戦略の生まれ変わる確率 1/50 より以下のように示すことができる。

$$keep= ((1-pc)*(1-0.02))$$

また、mh より今までの全ての行動を行って得たレバー報酬のなかで最大の値を maxP とする。Exploit するときの獲得コイン数の期待値には 1 ラウンド分の keep がかかり、observe するときの期待値には 2 ラウンド分の keep がかかるため、それぞれの期待値は以下の式で計算することができる。

$$obsP = \frac{Om*keep}{maxP*(1-keep)}$$

$$expP = \frac{bestpayoff}{maxP*(1-keep)}$$

この 2 つの値を比べることでどちらの方が効率よくコインを獲得できる行動か決定できる。

ただし、以下の条件に当てはまる場合では、異なる行動決定を行っている。

条件：pc < 0.05　かつ　20 回以上同じ行動を続けている場合。

1 ラウンドで得られるコイン数の平均 Rmean を用いて行動決定を行う。
Rmean+3 と bestpayoff を比べ、Rmean+3 の方が大きい場合は observe を行い、bestpayoff の方が大きい場合は exploit を行う。

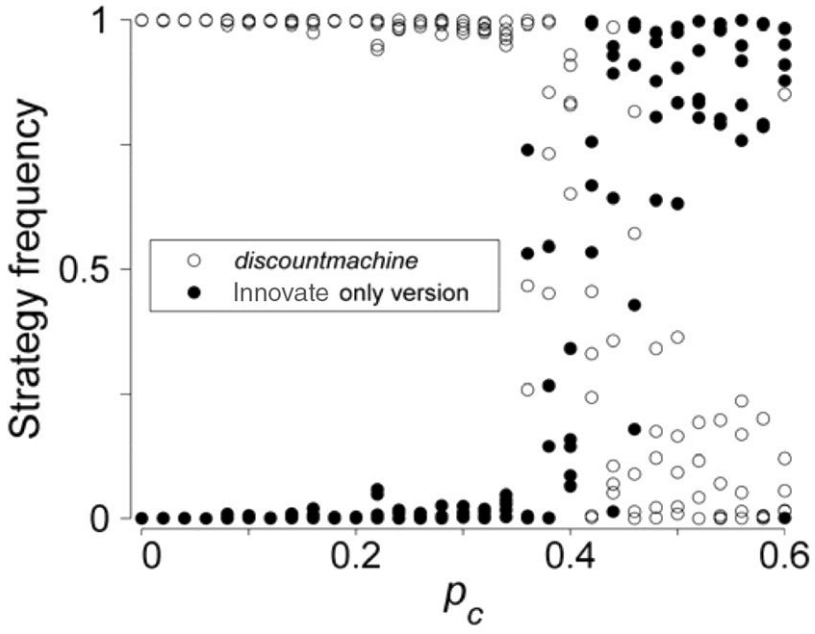
※この discount マシンでは、ニューラル・ネットワークを用いて行動を決定した場合についても記述してあるが、上記の場合と比べ、行動決定の結果に特別大きな違いは見られなかった。

以上を 1 ラウンドごとに繰り返し、より正確な情報で行動決定を行っている。

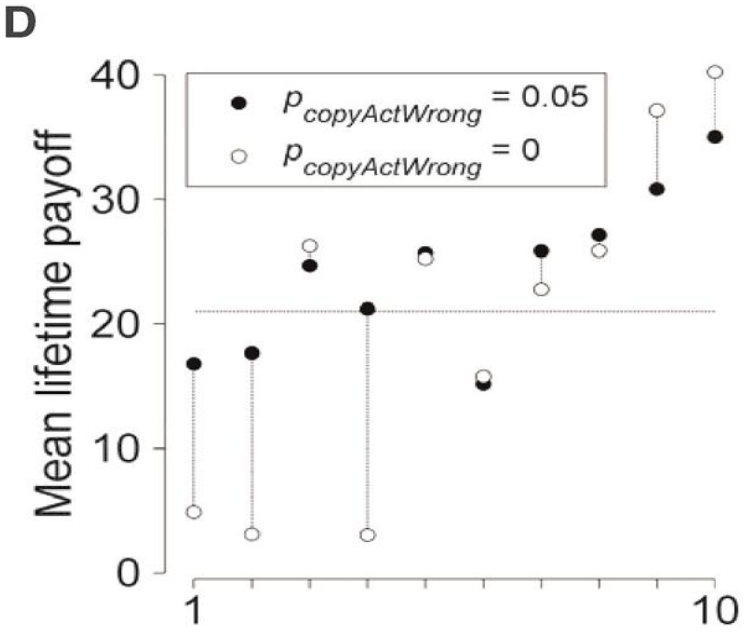
６　アルゴリズムの考察・改善点

トーナメントにおいて最もいい成績を収めたアルゴリズムは各レバーの獲得コインの期待値を計算し、その値をより正確に求めていくことで observe か exploit どちらの行動のほうが効率的にコインを獲得できるかを決定していくものであった。つまり Social Learning を行うためには周りの環境や自分の持つ情報を正確に捉え、それに応じて判断することが必要不可欠である。他の戦略を見ても今回のトーナメントでは、今までの自分の行動より、自分を取り巻く環境である pc の値や得られるレバー情報の期待値 mean などをより高い精度で計算し、それをもとに適切な行動決定ができている戦略が上位を占めていることがわかる。

逆に、discount マシンの意思決定について少々問題点がある。Discount マシンと observe の代わりに innvate のみを行うマシンとでトーナメントを行った場合、pc がある一定の値以上のとき後者のほうがいい成績を残した。



また各戦略のみの集団でトーナメントを行った場合の成績と通常のトーナメントの成績を比べると、順位がほぼ逆転した。



つまり、observe ばかり行う集団では discount マシンは強いわけではないということである。

これらより、意思決定に関しては、他の戦略や pc に応じて innovate を含む 3 つすべての行動について期待値を計算し、行う必要があるとわかった。こうした改良をすることでより効率の良い social learning のアルゴリズムとなるだろう。

参考文献

Why Copy Others? Insights from the Social Learning Strategies Tournament
(L. Rendell[1,*], R. Boyd[2], D. Cownden[3], M. Enquist[4,5], K. Eriksson[5,6], M. W. Feldman[7], L. Fogarty[1], S. Ghirlanda[5,8],T. Lillicrap[9], K. N. Laland[1,*])(16 November 2009.)