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

非線形物理学研究室　sp10104　大越創真

1、研究目的

「最も効率の良い学習方法は sorcial 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、トーナメントの概略

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、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、レバーの情報を調べる innovate、他人の引いた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た戦略の勝利となる。このトーナメントでは、innovate をほぼ行わず、exploit と observe のみ行っていた discount マシンが最もいい成績を収めた。

3、discount マシンのアルゴリズムの解析

実際に使われた discount マシンのプログラムコードを解析し、どのようなことが行われているか調べる。

まず、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 myhistory 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 myrepertorie に記録する。

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

このアルゴリズムは、各レバーに対して exploit した場合の獲得コインの期待値 hatbestpayoff 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 について求める。また、各戦略の経過したラウンド数を roundsalive とする。

myhistory、myrepertorie よりレバー番号ごとに情報を抜き出し、また observe しているときの情報は除外する。そのときのレバーの情報の推移からレバーの情報が変化した回数や変化した直後の値などを割り出す。それを用いて突然変異の確率 pc とレバーの平均コイン数 mean についてラウンドごとに以下の式で推定する。

$$\text{pc} = \frac{\textit{各レバーの変化した回数}}{\textit{各レバーの}\boldsymbol{roundsalive}\textit{の合計}}$$

$$\text{mean} = \frac{\textit{各レバーの初期値}+\textit{各レバーの変化した直後の値}}{\textit{レバーの数}+\textit{各レバーの変化回数}} = \frac{\textit{ラウンドごとの報酬}}{\textit{ラウンド数}}$$

平均コイン数 mean について、下の式はラウンド数が 6 より小さい場合、計算に使えるデータ数が極端少なくなってしまうため、今までのすべてのデータを使い精度をあげている。次に myhistory より observe しているときのデータを抜き出しその時の獲得したレバー情報の平均値 observablemean を計算する。この時、もし myhistory に observe しているデータがない場合は mean で代用する。

次に myhistory、myrepertorie よりレバー番号ごとに情報を抜き出す。そのレバーに対して行った最後の行動が observe だった場合、その前に 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 rounds に代入しておく。抜き出したデータの各列に対して下記の計算を行う。

payoff　= レバー報酬*((1-pc)**(roundsalive-ラウンド数))

noize = (1-pc)**(rounds-ラウンド数)

各列の payoff の合計を noize の合計で割ったものがそのレバー情報となる。最後の行動が observe 以外だった場合は最後のレバー情報を使う。各レバーが最後に使われたラウンド数から今のラウンド数まで経過したラウンド数を T として、レバーごとの獲得コインの期待値 bestpayoff をラウンドごとに以下の式で推定する。

bestpayoff = (1-(1-pc)**T)*mean+(((1-pc)**T)*レバー情報)

この計算で最も高い獲得コインの期待値だったレバーのレバー番号を hatbestmove、その期待値を hatbestpayoff にそれぞれ代入する。

最終的にどのような行動をするかについて pc を用いた以下の式で算出される値 discountfactor を用いて判断する。

discountfactor = ((1-pc)*(1-0.02))

この値と今までの最大獲得コイン数 maxpayoff などを用いて、observe したときの獲得コインの期待値と現在のレバー情報で exploit したときの獲得コインの期待値について下記の式で計算する。

$$\text{observablepayoff} = \frac{\text{observablemean}*\text{discountfactor}}{\text{maxpayoff}*(1-\text{discountfactor})}$$

$$\text{exploitablepayoff} = \frac{\text{hatbestpayoff}}{\text{maxpayoff}*(1-\text{discountfactor})}$$

もし pc が 0.05 未満で 20 回以上同じ行動をしている場合、myhistory より今まで

$$\text{meanpayoff} = 3 + \frac{\textit{exploitの合計獲得コイン数}}{\textit{roundsalive}}$$

の 1 ラウンド平均獲得コイン数 meanpayoff を計算し、hatbestpayoff と比べる。

この結果より、meanpayoff > hatbestpayoff ならば observe を行い、meanpayoff < hatbestpayoff ならば hatbestmove を行う。それ以外でもし pc が 0.075 未満で各レバーの roundsalive の合計が 12 より大きいとき、observe によって一度に得られる情報の数 nobs > 3 ならば observablepayoff と exploitablepayoff を比べる。observablepayoff > exploitablepayoff ならば observe を行い、observablepayoff < exploitablepayoff ならば hatbestmove を行う。それ以外は hatvestmove を行う。上記を繰り返していくことで、より正確な情報をもとに理想的な行動をしている。

4、アルゴリズムの考察・課題

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 discount マシンの集団でトーナメントを行った場合の discount マシンの成績と通常のトーナメントの discount マシンの成績を比べると、後者のほうがいい成績を残した。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

1) 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[1,5]**)(16 November 2009.)