G-23

環境のビット列表現による強化学習アルゴリズムの高速化について

Make learning faster at Reinforcement Learning by bit sequence representation of environment

磯村 厚誌† **大囿 忠親**† **新谷 虎松**†
Atsushi Isomura Tadachika Ozono Toramatsu Shintani

１．はじめに

強化学習とは，報酬と呼ばれる値を手がかりに，学習の主体であるエージェントが試行錯誤により学習を行う教師なし学習である．エージェントは，環境との相互作用をくり返すことにより，環境に適応した行動を学習して行く．

しかし，満足な学習結果を得るためには多くの繰り返しを要する．また，認識すべき対象が増加すると，エージェントが経験する可能性がある状態数が爆発的に増加する[2]．このため，強化学習における学習の過程には多大な時間を費やす場合がほとんどである．しかし，学習の過程の大部分はエージェントと環境の相互作用の繰り返しであるため，エージェントの，感覚入力に対する反応速度が向上させることができれば，学習のための計算時間を短縮できる．

本稿では，強化学習におけるエージェントの，感覚入力に対する応答速度の向上を目的とし，その評価の方法として，学習の過程全体での処理速度の比較を行う．

２．環境のビット列表現

エージェントは，取得した感覚入力に応じて，過去に同じ環境に遭遇していたのなら，それまで更新さてきた行動の重みにより，現在の行動を選択する．そのため，その時点における環境が，過去に経験した環境であるかを確認する必要がある．さらに，強化学習で扱う状態数は膨大な数になる場合がほとんどである．よって，経験した環境と現在の環境との照合過程における計算の負荷も大きなものとなる．本稿では，この状態の照合の過程を高速化することにより，エージェントの環境に対する反応速度の向上をはかる．また，その具体的な手法として，環境から取得した感覚入力をビット列による表現に変換する方法を考える．感覚入力をビット列で表現し，状態の照合のためにハッシュ関数を用いて計算速度の向上をはかる．

感覚入力に対するエージェントの反応速度を比較するための具体的な問題例として，倉庫管理問題を考える．本稿で扱う倉庫管理問題は，以下のようなルールで定義されるものとし，その盤面の例を図１に示す．

荷物は空白及び倉庫の方向に押すことのみ可能．荷物を荷物で押すことは出来ない．

- 荷物は空白及び倉庫の方向に押すことのみ可能．荷物を荷物で押すことは出来ない．
- 盤面の両端はループしている．例えば，左端からさらに左に移動したオブジェクトの移動先は，同じ行の右端になる．
- すべての荷物を倉庫へ運んだ状態を目標状態とする．

†名古屋工業大学　知能情報システム学科

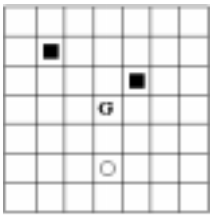

図１．倉庫管理問題の盤面

感覚入力として，盤面全体の各座標におけるオブジェクトの情報(空白，荷物，学習者及び倉庫)を与える．ここで，ハッシュ関数を用いるために，感覚入力を long 型の数値に変換する．この過程を以下に示す．また，以降排他的論理和を得るための演算は XOR であらわす．

入力：感覚入力 P
出力：P に対応するビット列 L

1. 初期パラメータとして， count = 0， long 値 L= 0 ，temp = 0 を用いる
2. P から座標(1,1)の情報を取り出し， p とする
   if p != NULL then
      if count < 32 then
         L を 1 ビット左シフトする
         if p!= 空白 then
            L の末尾 1 ビットに”1”をセットする
         count=count+1 としてステップ 2 をくり返す
      else
         if temp != 0 then
            temp = L XOR temp とする
         else
            temp = L とする
         L = 0, count = 0, p が P の最後の要素なら p = NULL ，それ以外なら次の座標の値として，ステップ 2 をくり返す
   if p == NULL then
      L = L XOR temp とし，出力する

感覚入力において変換される値は， long 型で定義しても，正負をあらわす最上位ビットを除いた 31 ビットまでなので，これを超える場合には残りの部分との排他的論理和を求めている．この値は、例えば図１のような盤面では、

```
(00000000100000000010000001000000)
 XOR          (00000010000000000)
=(00000000100000000010100001000000)
= 4,199,488
```

となる．こうすることで，ビット列から環境を一意に判断することが出来なくなるので，変換する以前の感覚入力も保持しておく．

このようにして得られた値に対して，適当な素数との剰余演算を行い，ハッシュ値を求める．このハッシュ値で参照される過去の感覚入力のリストと，現在の感覚入力とを比較し，同じものがあればその入力に対する各行動の重みを得る．過去同じ入力を得たことが無ければ，現在の入力を新たにリストに追加し，行動の重みの初期値を得る．

過去の感覚入力との照合を以上のように行うことで，エージェントはある時点での環境における行動の重みを即座に得ることができ，環境に対する反応時間を短縮することになる．

3．評価

前章で示した倉庫管理問題に対し，ハッシュ関数を用いた場合とそうでない場合との比較を，状態空間の広さを変えて行った．また，ハッシュ値を得るための素数の大きさによる効果の比較も行った．

それぞれのケースでの処理速度の比較は，倉庫管理問題解決プログラムの実行時間によって比較する．だたし，プログラムの終了条件は，エージェントの選択する一連の行動が収束した時点では無く， 10,000 回の試行をこえた時点で終了とする．これは，速度の比較を行う際，収束するまでにかかる時間が現実的ではないケースを含むためである．また，実験で得られた処理速度の値は， CPU:Pentium4 1.5GHz, Memory:512MB の Linux マシンにおける値である．

3.1 状態空間の広さによる影響

ハッシュ関数を用いない場合との比較には，状態空間の広さによる影響の差も比較するために，荷物を 1 つしか置かない問題と， 2 つ置く問題を使用する．図 2 に盤面の機能別の実行時間に関するグラフを示す．グラフ中の実線が，本稿で提案したハッシュ関数を用いたアルゴリズムによるグラフを表し，破線は用いていない既存のアルゴリズムによる結果を表す．横軸には盤面の広さをとり，図の左のグラフが荷物 1 つの場合を，右が 2 つの場合を表す．また，ハッシュ関数を用いる場合には，ハッシュ値を得るために用いる素数として， 2111 を使用する．

荷物を 1 つだけ置く場合(図 2 左グラフ)には， 7×7 の盤面で，出口を固定したとしても管理人と荷物の配置パターンだけで， 48×47 = 2256 の状態をとり得る．これに対し，荷物を 2 つ置く問題(図 2 右グラフ)では， 1 つ目の荷物が出口に運ばれた後の状態も考えると， 49× 48×47=110544 となる．同様に， 4×4 の場合には 3360 となり， 7×7 の盤面に荷物を 1 つとする場合の状態数を上回る．そこで，荷物が 1 つの場合と 2 つの場合の問題として，それぞれ盤面の広さが 4×4 ～ 7×7 のものを用いる

図 2 に示すように，既存のアルゴリズムでは，状態数の増加によって計算時間が大幅に増加することがわかる．特に認識すべき対象が増えてからでは，盤面の拡張に対する計算時間の増加の割合が非常に大きい．このため，既存のアルゴリズムに基づく実行時間では， 6×6 以降は結果を取得していない．これに対し，本稿で提案したハッシュ関数を用いたアルゴリズムの実行時間は，増加の割合がどちらも非常に低くなっている．このため，たとえば 5×5 の盤面で比較すると， 50 倍程度効率が上昇するという結果が得られている．本アプローチで特筆すべき点は，図 2 で示すように，盤面が 6×6 以上の場合でも，実際的な時間で高速に実行結果を得ていることである．

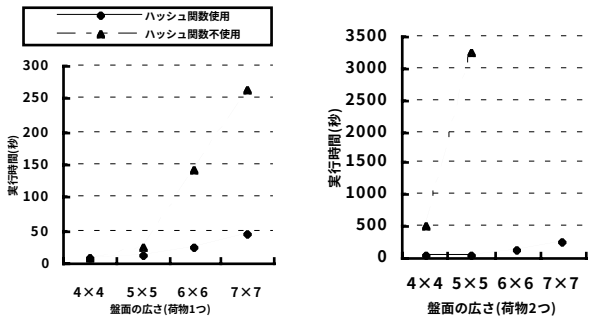

図 2．盤面の構造別の実行時間

3.2 素数の大きさによる影響

ハッシュ値を得るための素数の，大きさ毎の計算速度の比較は，盤面の広さが 7×7 で荷物が 2 個の問題で行った．

用いる素数が非常に小さい場合には，かなりの計算時間を必要とするが，大きさを変えた際の，計算時間が減少する割合が大きい．これに対し，ある程度の大きさの素数を用いると，それ以降向上の幅は小さくなる．例えば，状態数が 110544 である 7×7 の問題に対し，ハッシュ値を得るための素数として 2111 と 49999 を用いた場合の違いはわずかであった．また， 13 と 29 を用いた場合の差は倍近くになった．このとき用いた素数は， 7 から順に， 13 ，29 ，・・・と，順に倍程度になるものを使用した．結果，変化が少なくなり始めたのは 251 を使用した場合であった．約 8 倍の 2111 と比べると， 1.2 倍ほどの効果に収まる．よって，ハッシュ値を求めるために用いる素数は，環境がとり得る状態数に比べてかなり低くても良いことがわかる．ただし，大きい素数を使用することで遅くなることは無かったので，余裕のある値が望ましい．

4．まとめ

強化学習の学習過程において，感覚入力の照合の過程を高速化することで，エージェントの入力に対する反応の速度の向上を試みた．特に，複雑な問題に対しては，ハッシュ関数を用いることで，全体の処理を通して，単純に照合を行う場合の数十倍の高速化を実現することが出来た．学習の過程は，エージェントと環境の相互作用の膨大な繰り返しであるので，この結果は，エージェントの環境に対する反応の速度が大幅に向上していることを示している．

今後の課題としては，感覚入力からビット列への変換を最適に行うことが挙げられる．このためには，変換のための計算負荷についても考慮する必要がある．また，今回高速化されたのはエージェントの反応の速度であるため，状態数が増加した際に伴う，収束に必要になる試行回数の増加を抑えることは出来ない．しかし，全体の計算速度を向上させることが出来た為，扱うことのできる問題の範囲をかなり拡張することが出来た．

参考文献

[1] Sutton, R.S. and Barto, A, “Reinforcement Learning : An introduction, “ A Broadford Book, The MIT Press 1998.

[2] 荒井幸代, ”マルチエージェント強化学習～実用化に向けての課題・理論・諸技術との融合, ” 人口知能学会誌, Vol.16, No.4, pp.467-481, 2001.