Rep. 寺井 仁（'04/06/15）

DUALITY of the MIND

Chapter 3: Current Implementations

Ron Sun

この章では， CLARION モデルのアブストラクトバージョンのアルゴリズム的実装について述べる．特に，CLARION のインプリメンテーションにおける主要な側面において，詳細に見てゆく．

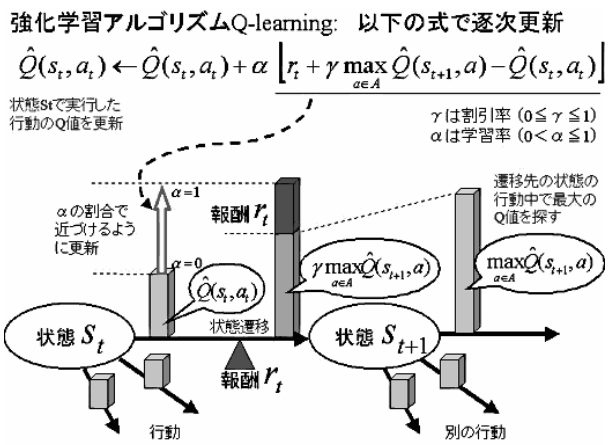
■■モデルの動作

- CLARION はトップ/ボトムの 2 つのレベルからなる
- 2 つのオペレーションの全体的な動作
  0. 現在の状態 x を観察する
  1. **ボトムレベル**で，状態 x において可能な行動 a の価値を計算する
     Q(x, a1), Q(x, a2),...
  2. 現在の状態 x(ボトムレベルからあがってくる)とマッチする**トップレベル**のルールから，可能な行動のすべて(b1, b2,...)を見つける
  3. トップレベルの値（aj）とボトムレベル値(bj)から，**適切なアクション b を選択**する
  4. アクション b の実行，次の状態 y を観察，可能なら強化 r
  5. Q-learning バックプロパゲーションにもとづきボトムレベルをアップデートする
  6. **Rule-Extraction-Revision** によりトップレベルのルールをアップデートする
  7. ステップ 0 へ

■■強化学習（ボトムレベルの実装について）

□Q 値

- ある状態における行動の"quality"の価値
  Q(x, a): 状態 x における行動 a の望ましさ
  状態 x は知覚入力によって得られる
  Q 値によって行動を決定する
- 行動の決定方法
  現在の状態における最大の Q 値を持つ行動を行う（グリーディ方策，ε グリーディ方策）
  ある確率で異なる行動を試す（ボルツマン選択，ルーレット選択等）
- Q 値の獲得方法
  Q-learning アルゴリズム（Watkins 1989）
  → 累積報酬の最大値を見積もる
  Q(x, a)の更新は，現在の状態と選択された行動を評価した結果の差分によって行う
  $\Delta(x, a) = \alpha(r + \gamma \max Q(y, a) - Q(x, a))$
- 連続的な Q 値のアップデートを通して，エージェントは先の行動を考慮し学習を行う



□反応ルーチン

・ 強化学習を通して成長した反応ルーチンは,
　－顕在的な（シンボリックな）プランニングなしに
　－その時々の環境からの入力を用い
　一連の行動（逐次的意思決定）を行う

□Q-learning の実装方法

・ 4 層のネットワークからなる
　３層ネットワーク：
　　Q 値を求めるためのバックプロパゲーションネットワーク
　３層目（バックプロパゲーションネットワークの出力層）：
　　各行動の Q 値を示す
　４層目：
　　確率的（ボルツマン選択）な意思決定を行う
　　$p(a \mid x) = \exp(1/\alpha \ \ Q(s, a))/\Sigma e(1/\alpha \ \ Q(x, a_i))$
・ なぜネットワーク表現を用いるのか？
　通常 Q-learning では状態・行動対に対する Q 値のテーブル（ルックアップテーブル）を用いるが,
　この方法では，状態数が膨大になるため Q 値の見積もりにバックプロパゲーションネットワークを
　用いる.
・ バックプロパゲーションによる構造的信用
　エージェントはある状態においてどのエレメントが，信用と責任を与えられるかを知る
・ Q-learning による時間的信用
　エージェントは強化を受けることにより，どの行動が成功または失敗を導くかを知る
　→ Q-learning とバックプロパゲーションの組み合わせが，継続的な絶え間ない世界におけるエー
　　ジェントの探索のみを基礎に，潜在的知識の構築を可能にする.
　　※外的な教師やア・プリオリな領域固有の知識を必要としない.

■■ルールの抽出（ボトムレベルからトップレベルへの学習）

・ Rule-Extraction-Revision
　トップレベルにおける新規なルールの学習アルゴリズム
・ 各ステップにおいてルールの学習（抽出と更新）が行われる
・ ルールの抽出
　ボトムレベルにおいてある行動が決定されたら，エージェントはその決定に対応するルールを抽出
　し，ルールネットワークに追加する
　　状態 → 行動
　状態：感覚入力の各エレメントの論理積
・ ルールの検証
　－ルールの縮小
　　結果が失敗だったとき，現在の事例を除き，ルールはより特殊なものにする

－ルールの拡張

結果が成功だったとき，ルールをより一般的なものにする

・ 情報利得

各ステップにおいて，行動 a に関して，ルールの条件とその条件にプラス/マイナス１したマイナーバリエーションに対して情報利得を算出する

$$IG(A, B) = \text{・・・}$$

$PM_a(A)$（ポジティブマッチ）：

状態 A のときに行動 a を行って結果がポジティブだったときの数

$NM_a(A)$（ネガティブマッチ）：

状態 A のときに行動 a を行って結果がネガティブだったときの数

※ポジティブ/ネガティブは Bellman residual(Q 値のアップデート量)で決まる

$r + \gamma \max_b Q(y, b) - Q(x, a)$

※※A,B は同じ行動 a を導く異なる状態

→ B よりも A が良い状態なら IG は正の値となる

（この手法は一般的であり, 例えば, 多くの Inductive Logic Programming System で用いられている.）

□ルールの抽出

もし,

$r + \gamma \max_b Q(y, b) - Q(x, a) > threshold$

&

トップレベルにこのステップをカバーするルールがない

なら,

ルール：C → a

C：全入力値

a：行動

を生成する.

□ルールの拡張

もし,

$IG(C, all) > threshold1$

&

$\max_{C'} IG(C', C) \geqq 0$

なら,

（もし現在のルールがうまく機能しており，拡張されたルールがよりよいものなら）

$C'' = \text{argmax}_{C'} IG(C', C)$

の新しい（拡張された）状態をルールの条件とする.

（C：現在の条件，all：すべての条件，C’：修正された条件（C＋１））

□ルールの縮小

もし，

IG(C, all) < threshold2

$\max_{C'}$IG(C', C) > 0

なら，

（もし現在のルールがうまく機能せず，縮小されたルールがよりよいものなら）

C" = $\text{argmax}_{C'}$IG(C', C)

の新しい（縮小された）状態をルールの条件とする．

もし条件の縮小がルールに対してどのような入力状態にもマッチしなければ，ルールを削除する．

（C：現在の条件，all：すべての条件，C'：修正された条件（C－１））

・ 拡張の間，拡張されたルールによってカバーされた既存のルールはその子リストに含まれれ，不活性化されるが，ルールが削除または縮小されたときに，子リストのルールが再活性される

・ ルールのネットワークによる表現
  －ルールの条件において条件要素を表現しているノードが，結果を表現しているノードにリンクしている．
  －もし条件要素がポジティブなら，ボトムアップリンクはポジティブウェイト w を，そうでなければ，ネガティブウェイト-w をもつ．

・ ルールの抽出に対するパフォーマンスの優位性
  －ルールの抽出は学習を速める
  －新しい状況の学習をガイドする
  －ルールは学習された知識の伝達を助ける（転移）
    → Appendix
  －以下の要因がモデルのパフォーマンスの優位性を生み出している
  （１） ２つのレベルの相補的な表現：離散的 vs. 連続的
  （２） ２つのレベルの相補的な学習プロセス：one-shot ルール学習 vs. 漸進的 Q 値の見積もり
  （３） 適切なルール学習の基準の使用

■■価値関数とルールの統合（ボトムとトップの統合）

・ 最終的にボトム/トップどちらの行動を採用するのかに対して，結果を統合する２つの方法を用いる

□percentage method

・ ランダムに p パーセント，少なくとも１つのルールが現在の状態に合致すれば，トップレベル（ルール）からの結果を用いる
・ それ以外は，ボトムレベルの結果を用いる（ボトムレベルは常に存在する）
・ トップレベルからの結果を使うとき，トップルールが複数マッチしていればランダムに選択
・ ボトムレベルからの結果を使うとき，Q-learning の確率的行動選択を行う

□stochastic method

・ 加重和によって各アクションに対する一致度を計算する

もし,

トップレベルが行動 a の活性値 v_a を持っている（ルールなのでバイナリ，0 or 1）

&

ボトムレベルが行動 a の活性値 q_a（a の Q 値）を持っている

ならば，最終的な出力は,

$w_1 \times v_a + w_2 \times q_a$

となる.

・ 加重和を基礎にした確率的意思決定は，すべての可能な行動の中から行動を選択する

percentage method はある確率でトップレベルでマッチした中からランダムに実行する．その際トップレベルにルールがなければ，ボトムレベルから確率的選択を行う，というようにトップ/ボトムどちらかからの行動を実行する．一方，stochastic method はトップ・ボトムのすべての可能性を考慮している.

■■プランの抽出

・ ルールの抽出は各状態に注目し個別のルールを生成する方法であるが，シーケンシャルなタスクを達成するためにルールを連鎖させてはいない

→ 連続的なタスクを達成するための顕在的プランを抽出する

・ 顕在的プランは，特に高次レベルの認知スキルの獲得において繰り返し実証されてきた(VanLehn 1995, Pfleger and Hayes-Roth 1997, Sacerdoti 1974)

・ プランの抽出法

学習された Q 値をプランに変化させる

・ ゴールに達するベストな行動の系列を見つけるために Q 値を用いてビームサーチを行う

アルゴリズム：

ビーム幅 n に対する初期値をセット

ビーム幅 m に対する初期値をセット

探索の深さ D に対する初期値をセット

p(G) > δ（もしくは，時間制限（ア・プリオリに決定)）まで繰り替えす

n, m, D を用いて**ビームサーチ**を行う（つまりプラン抽出アルゴリズムを呼ぶ）

探索の結果から p(G)を計算する

$p(G) = \Sigma p(x')$

（x' = G, 異なるパスからのおのおのの結果）

n = f1(n)(つまり，f1(n) = n + 1)

m = f2(m)(つまり，f2(m) = m + 1)

D = f3(D)(つまり，f3(D) = D*2)

ビームサーチアルゴリズム：

（１） current state sets（ビームサーチによる軌跡を保持）：

CSS = {(x0, 1)} and CSS' = { }（状態，その状態になる確率）

（2） 終端に達するか，条件（step > D）を満足するまで（2）～（9）を繰り返し
（3） 現在の状態から可能な次の状態すべてに対する遷移確率を計算する
（4） 各行動に対して，現在の状態から行動を行った後ゴールに到達するか（utility：有用性）の確率を計算する（行動後の最大の Q 値をもとにする）
（5） 各行動に対して，CSS'におけるすべての状態に対して utility を計算する
（6） CSS における各状態 x に対して，最も高い utility を持つ行動 ux を選択する
（7） CSS'におけるすべての状態について, もっと良いデフォルトの行動 u（utility が最大の行動）を選択する
（8） n 個の最も高い確率を持つ n 個の状態を含むように，CSS をアップデートする
（9） m 個の最も高い確率を持つ m 個の状態を含むように，CSS'をアップデートする

・ このアルゴリズムを用いて，もっとも成功するだろう一番良い行動を選択する

・ プラン抽出のアドバンテージ
その時々のセンシングに頼らなければならない closed-loop（出力信号を入力側にフィードバックする）方策の代わりに，抽出されたプランは open-loop（出力信号を入力側にフィードバックしない）で使用される
→ フィードバックがない場合や信頼できない場合に有効
確かなフィードバックが存在したとしても，センシングコストの節約の観点から有効
・ 実験から抽出されたプランが“方策の圧縮”を導くことができた
－方策の記憶コストを抑えられる
－抽出されたプランは，サブゴールのセットアップや，探索の制約，そして他の処理を通して，より良い系列の発見や，環境の変化への適応など，更なる学習を助けるために利用可能である

・ ゴールへの到達の確率に関しては信頼ができ, 完全なアルゴリズムである（Sun and Sessions 1998）
・ プランの抽出の有用性は実例によって保障されている.
→ Appendix

□Appendix: 基本的な CLARION モデルのパフォーマンス（迷路探索）

・ 課題
迷路 → Fig 3.4
・ エージェントのインプットセンサー
左，前，右；壁あり，壁なし，ゴール
場所に関する情報なし，場所の方向に関する知識なし，ゴールの場所に関する情報なし
・ 動き
前進，左にターン，右にターン
・ 報酬
ゴール：１の報酬，壁に激突：-0.1 の罰

このケースでは状態数が少ないため Q-learning のルックアップテーブルを使用できるが，CLARION は迷路探索に対してデザインされていないため，使用しなかった.（例えば，前に紹介した機雷原探索

では状態数がインプットが４１あり，状態数が 10^{12} も存在するためルックアップテーブルは使えない）

学習スピード

- CLARION をボトムレベル（Q-learning）のシステムとの比較
  → Fig 3.5
  60 エピソード後の結果
  Perc.x: ルールの使用に関して percentage method を用いる
  Stoc.x: ルールの使用に関して stochastic method を用いる
  .gen: expansion と shrinking を行っている（それ以外は無視）
- ルールが適度に使われたときが，ボトムレベルの学習だけよりも早く学習が進む.
  （例：Perc.80, Stoc.20）（t test $p<0.01$）
- 学習されたルールの数は同程度である．（7 前後）
- 学習曲線
  → Fig 3.6

トレーニングされたパフォーマンス

- 60 エピソード後の１エピソードにおける平均ステップ数
  → Fig 3.7
  CLARION は Q-learning のみの学習よりも効率がよい
  （R-moves：トップレベルのみ， Q-moves：ボトムレベルのみ）
  → トップレベルとボトムレベルの相互作用が確認できる
  CLARION がボトムまたはトップレベルのみよりもよりよいパフォーマンスを示す
- トップレベルからボトムレベルへの干渉
  Fig3.7 を縦に見ると，トップレベルの学習が全体のパフォーマンスを向上するだけではなく，ボトムレベルそれ自体を向上させていることがわかる
  → ルールはボトムレベルかが学習されるが，また，それがボトムレベルを促進させる

転移

- 小さな迷路で 60 エピソード学習後大きな迷路(Fig 3.8)に適用
  → Fig 3.9
  CLARION の転移がボトムレベルのみのシステムよりもより良い
  R-moves は Q-moves 単独または CLARION モデル全体よりもしばしば良い結果を示す
  R-moves の優位性は，ルールの学習が新規でより複雑な環境への転移を促進することを示す

□Appendix: ルール抽出の例

迷路課題において学習されたルールの例

- もし左のコーナーなら，右に回る
- もし右にいけるなら，左に回る
- もし行き止まりなら，左に回る
- もし前方が開けているなら，前に進む

・ もしターゲットが右にあるなら，右に回る
・ もし回廊なら，前に進む
・ もしターゲットが前にあれば，前に進む
→ 迷路の探索に対する良いルールセットであることが簡単にわかる

機雷原ナビゲーションにおいて学習されたルールの例

方位：まっすぐ
一番密集していない場所：真ん中
一番遠くの機雷：真ん中
左の平均機雷距離：近く
中心の平均機雷距離：とても遠く
右の平均機雷距離：遠く
方向：まっすぐ，スピード：とても速く
・・・

□Appendix: プラン抽出の例

Tower of Hanoi

・ スタート状態：ペグ 1 に 3 つのディスクがある
・ ゴール状態：ペグ 3 に 3 つのディスクがある
・ 可能な行動：Fig 3.15
・ 可能な状態：Fig 3.16
・ 状態遷移：Fig 3.17

・ closed-loop 方策を学習するために Q-learning を適用
・ 100%の確率で抽出されたプランによってゴールに到達
・ Q 値から抽出された非条件付プラン：
Step1: ペグ１から３にディスクを移動（アクション１）
Step2: ペグ１から２にディスクを移動（アクション０）
Step3: ペグ３から２にディスクを移動（アクション５）
Step4: ペグ１から３にディスクを移動（アクション１）
Step5: ペグ２から１にディスクを移動（アクション２）
Step6: ペグ２から３にディスクを移動（アクション３）
Step7: ペグ１から３にディスクを移動（アクション１）

■■まとめ

・ この章では，CLARION モデルの実装について議論した
・ 以上のアルゴリズムは強化学習を用いたボトムレベルの学習，ルールやプランの抽出を用いたトップレベルの学習をカバーしている
・ 注意点

－以上のアルゴリズムが最終的なものではない
－多くの実験を行い合理的な実装になったとはいえ，なお，よりよくするための余地が残っている
－機械学習はまだ未成熟で発展している分野であり，CLARION を改善するためのさまざまなタイプのアルゴリズムを取り入れることができるだろう