2005/03/09 寺井

Adapting to the task environment: Explorations in expected value

Wayne D. Gray, Michael J. Schoelles, Chris R. Sims

Cognitive Systems Research, in press

■Abstract

- ストラテジ選択におけるトレードオフ
  「どのようにタスクがデザインされているか」における小さな違いが原因
- 競合するストラテジ（プロダクション）の選択をモデル化
  - ACT-R のデフォルトの選択メカニズム（期待値式）を使い，トレードオフがうまくモデル化できないことを示す
    ※ACT-R の選択メカニズムは，多くの成功を収めている（see, for example Anderson, J. R., & Lebiere, C., 1998）
  - 選択メカニズムを改良したモデルを示す
    - ACT-R デフォルト：すべての行動が「成功」または「失敗」という 2 値で処理される（モデル化に失敗した原因）
    - 改良したモデル：「成功」/「失敗」の判断メカニズムを段階的なメカニズムに変更（多くの環境では，行動は「部分的な成功」と「部分的な失敗」の混ざった結果に遭遇する）

■Introduction

- タスクの解決におけるストラテジの選択
  - 新たなストラテジの発見を要求するような新規なタスクはほとんどない
  - 多くの状況では，すでに獲得されたストラテジのセットからどのストラテジがより現在のタスク環境に適応するかを学習
- 適応における 2 つの性質
  - 連続的（通常）
  - ガイドされない（しばしば）
- ストラテジの選択
  - ルーチン化されたタスク（例：本のコピー）であっても，「変化」と「発展」を続ける（Agre & shrager, 1990）
  - 説明的な手引きがない（教師なし）にもかかわらず変化が起こる
- 教師なし学習
  - 一般的に，非局所的蓄積効果モデル（non-local cumulative-effects models; Davis, Staddon, Machado, & Palmer, 1993）は，教師なし学習（Sutton & Barto, 1998）を必要とする
  - ほとんどのニューラルネットワークでは教師あり学習が行われ，どのように振る舞うべきかが伝えられる（教師信号）
- 本論文の流れ
  1. タスク（ブロックワールド(**Fig. 1**)）における心理実験のデータを示す
  2. 期待値式(Anderson, et al., in press; Anderson & Lebiere, 1998)を用いたモデル
     - 期待値の更新のタイミング（実行後に更新，タスク終了時に更新）による変化
     - ACT-R における期待値式（プロダクションの選択メカニズム）がブロックワールドに対し

てなぜ不適切なのか

3. ACT-R の期待値式を変更した 3 つのモデル
   - ✧ 経験的なデータへの一致/不一致と同様，各変更点が各モデルの振る舞いに及ぼした影響

4 ACT-R の期待値式を用いた擬似的モデルの 2 つのバリエーションの結果を示す（デフォルトの ACT-R のパラーメータを心理実験の値をもとに設定）

5 まとめ

■ブロックワールド

□タスクの概要

- ブロックワールドタスク → **Fig. 1**
  - 「インタラクションストラテジ」(interaction-intensive strategy)と「記憶ストラテジ」(memory-intensive strategy)の間のトレードオフの研究に向いている(Ballard, Hayhoe, & Pelz, 1995; Fu & Gray, 2000; Gray & Fu, 2000; Gray, Sims Fu, & Schoelles, in preparation)
  - タスクの目的
    「ターゲットウィンドウ」内のブロックのパターンを,「リソースウィンドウ」内のブロックを使って,「ワークスペースウィンドウ」に再現する
- タスクの特徴
  - ターゲットウィンドウに 8 個の色のついたブロックがランダムに配置される
  - 3 つのウィンドウはグレーのウィンドウによってカバーされている
    - ✧ リソースウィンドウとワークスペースウィンドウ
      カーソルがウィンドウ内に入ったらすぐにカバーが消える
    - ✧ ターゲットウィンドウ
      カーソルがウィンドウ内に入ってから一定時間経過した後にカバーが消える
- 操作要因
  - コスト：ターゲットウィンドウのカバー開くまでの時間
    - ✧ Lockout 0：マウスが入ったら一瞬で開く
    - ✧ Lockout 400：マウスが入った後 400msec(6 2/3 秒)経過してから開く
    - ✧ Lockout 3200：マウスが入った後 3200msec(53 2/3 秒) 経過してから開く
- その他
  - 繰り返し回数：48 回（以下では試行数と呼ぶ）
  - タスクの終了：配置できるまで行う

□ストラテジ

- ターゲットウィンドウの情報へのアクセスにおける使用ストラテジ
  - インタラクションストラテジ
  - メモリストラテジ
- ストラテジの特徴
  - 極端なインタラクションストラテジ
    ブロックの「色情報」と「位置情報」を取得するために，そのつどターゲットウィンドウにアクセスする
  - 極端なメモリストラテジ
    一度にすべてのブロックの「色情報」と「位置情報」を取得する

- 予想
  - ターゲットウィンドウにおける情報へのアクセスのコストが上がると，インタラクティブストラテジからメモリストラテジへ移行する

□心理実験の結果 → Fig. 2

- 被験者：3 条件（Lockout 0, 400, 3200）に各 18 名ずつ
- 分析対象：25 から 48 試行（安定（学習）した状態）
- 最初にターゲットウィンドウを開いた後に正確に置くことができたブロック数を測定
- 結果
  - Lockout 時間の増加にしたがって，正確に置くことができたブロック数が増加
  - インタラクション優位のストラテジからメモリ優位のストラテジへと選択を変化させている

■最適なストラテジの選択の失敗: 信用割り当て（credit assignment）と期待値の問題

□モデルの詳細

エンコードストラテジ

- エンコード数
  - ターゲットウィンドウを開くコストが増加すると，ウィンドウにとどまる時間が増加
    → エンコード数の増加
- 複数のエンコードストラテジとタスクの解決
  - エンコードストラテジ
    Encode-1～Encode-8：ターゲットウィンドウを開いたときにエンコードするブロックの数
  - もっとも高い期待値を持つストラテジが選択され，エンコードされた記憶の検索を試みる
  - 検索可能な記憶エレメントがなくなったら，新しいエンコードストラテジを選択する
  - ワークスペースウィンドウに 8 つのブロックが正確に置かれたら試行を終了

期待値の算出

- 期待値の比較によってエンコードストラテジが選択される
- ACT-R の期待値式
  [期待値] = [成功確率][定数] – [コスト] ± [ノイズ]

[成功確率]： [成功数] / ([成功数] + [失敗数])
[成功数]：プロダクションが成功した総数
[失敗数]：プロダクションが失敗した総数
[定数]：単位時間で表現される定数，与えられたゴールをやり遂げるのにかかる時間
[コスト]：[費やした時間] / ([成功数] + [失敗数])
[費やした時間]：過去にこのプロダクションを実行して費やした時間の総数
[ノイズ]：期待値の変動性
小さなノイズ：初期の P や C に依存し，ストラテジの選択が初期に収束する
大きなノイズ：P や C に関係なく，すべてのストラテジが選択される
→ 教師なし学習において大きな役割を果たす（Sutton & Barto, 1998）

期待値式の更新が異なる 5 つのモデル

- デフォルトモデル
  - ✧ Vanilla-Once
  - ✧ Vanilla-Each
- 改良モデル

  Success-Weighted

  All-Weighted

  Mixed-Weighted

信用割り当て問題（credit assignment problem）

- 信用割り当て問題とは

  「成功」に対する信用を，「成功」にかかわった意思決定（選択されたプロダクション）にどのように配分すべきか？
- タスクの性質による問題
  - 「成功」の総数

    課題は「成功」して終了するため，各プロダクションは実行された回数分，「成功」の総数が増加する
  - 「費やした時間」の総数

    「費やした時間」＝「課題終了時の時間－プロダクション選択時の時間」であるため，タイミングによって評価が異なる
- 2 つの類似した基礎的な ACT-R モデルを用いて期待値の更新タイミングによる違いを比較する（以下，Vanilla モデルと呼ぶ）
  - Vanilla-Once：8 つのブロックをすべて置いた後に更新（1 試行毎）
  - Vanilla-Each：エンコードストラテジの発火後に更新（更新に用いる時間：エンコードストラテジを選択した時 ～ 検索可能な記憶エレメントがなくなるまで）

成功とコストの更新における問題

- Vanilla モデル（Vanilla-Each, Vanilla-Once）と実験データの比較
  - 人のデータを再現できない

    → Fig. 2（ターゲットウィンドウを最初に開いた後に置いたブロック数の変化）
    - ✧ 人はロックアウト時間が長くなると最初に置くブロック数が増加する

      （RMSE = 1.87 for Vanilla-Each, 1.46 for vanilla-Once, r^2 = 0.33 for Vanilla-Once, for Vanilla-Each）
    - ✧ Vanilla-Each： Encode-1 ストラテジを選択 → **Table 1**
    - ✧ Vanilla-Once： Encode-1, Encode-2 ストラテジを選択 → **Table 2**
- Vanilla モデルが実験データを説明できない原因
  - ACT-R の期待値算出が原因
  - [期待値] = **[成功確率]**[定数] – **[コスト]** ± [ノイズ]
    - ✧ [成功確率]：常に 1（終了時にはすべてのブロックが置かれて（成功して）いるため）
    - ✧ [コスト]：期待値はコストによって支配される
  - Vanilla-Each
    - ✧ 更新タイミング：エンコードストラテジの発火毎
    - ✧ コスト（ブロックのエンコード，リソースウィンドウからのブロックの取得，ワークスペ

ースウィンドウへのブロックの配置）
エンコード数多 > エンコード数少
→ より多くブロックを置くエンコードストラテジの期待値はすぐに低下する

- Vanilla-Once
  - ✧ 更新タイミング：試行の終わり
  - ✧ コスト
    - ACT-R のワーキングメモリの減衰機能により，Encode 数が多くなると完全な再生は行えない（Encode-8 では 5 つ程度しか再生できない）
    - Encode-8：Encode-8 自身のコスト＋不足分を補う Encode ストラテジのコスト
    - Encode-1：Encode-1 ストラテジ 8 回分のコストの平均値（(8 * 7* . . . * 1）/ 8)
      エンコード数多 > エンコード数少

□モデルの改良

- 実験データの質的特徴（**Fig.2** における右上がりの特徴）との不一致の原因
  - ACT-R のデフォルトメカニズムの不備（弱点）が原因
  - 現在のメカニズムは 2 値的フィードバックに制限されている(Fu & Anderson, 2004)
  - 具体例
    Encode-4 が選択され，4 つのブロックを正確におくことができた場合,「成功」として 1 回更新される.
    → 報酬の量（1 回の更新）が成果の量（4 つもブロックを置くことができた）を反映していないため，期待値はコストによって完全にドライブされる
    → 高い成果はコストが高いため，ACT-R はコストの低いエンコードストラテジでゴールを目指す
- 改良すべき点
  - 「成功」と「失敗」を段階的に定義
    期待値式を修正することによりデータの予測が可能？
    → 「3 つ」のモデルで検証

■「成功」と「失敗」の重み付け

- 置くことができたブロック数によって部分的な「成功」を定義する → **Table 3**
- 期待値の更新のタイミング：ストラテジの発火毎（Vanilla-Each モデルと同様）

□Success-Weighted モデル

- 「成功数」を変更
- 「エンコード」と「配置」のコストを正しく置くことができた数によって定義する
- [成功確率]

  [成功確率] = 「成功数」 / (「成功数」 + 「失敗数」)

  ↓

  [成功確率] = |「成功総」 / 「成功数」| = 1 ※常に 1
- [コスト]

  [コスト] = 「費やした時間」 / (「成功数」 + 「失敗数」)

  ↓

[コスト] = 「費やした時間」 / 「成功数」

→ 成功した数で評価される

例：Encode-5 で 3 つ正しく置くことができれば,「成功数」 = 3

□All-Weighted モデル

- 変更点
  - 「成功数」 と「失敗数」を段階的に定義
  - 「成功数」: 正しく置くことができたブロック数
  - 「失敗数」:「エンコード数」－「正しく置くことができたブロック数」
- [成功確率]

  [成功確率] = 「成功数」 / (「成功数」 + 「失敗数」)

  → エンコード数よりも検索できた数が少ないと「罰」を受ける
- [コスト]

  [コスト] = 「費やした時間」 / (「成功数」 + 「失敗数」)

  → 「成功数」と「失敗数」の比率に影響されない

  多くのブロックを置くほどコストが増加する（時間がかかるため）

  同じストラテジで，早く失敗したほうがコストが減る！

□4.3. Mixed-Weighted モデル

- 変更点
  - All-weighted から[コスト]の式を Success-Weighted のものに変更
- [成功確率]

  [成功確率] = 「成功数」 / (「成功数」 + 「失敗数」)

  → エンコード数よりも検索できた数が少ないと「罰」を受ける
- [コスト]

  [コスト] = 「費やした時間」 / 「成功数」

  → 成功した数でのみ評価される

■シミュレーション結果：重み付けスキーマの比較

□Success-Weighted モデル

- パフォーマンス → **Fig. 2**
  - Vanilla モデルと異なり，Success-Weighted モデルは一貫して人のパフォーマンスを上回った
  - 人のデータとの量的一致はほとんどない（RMSE = 1.71）
  - 人のデータとは質的かなり一致する（r^2 = 0.89）

    → しかし，これは単に，値が大きくなったことに起因する
  - グラフの形状（傾きを指標とする）の比較では 4%しかマッチしない
- 各ストラテジの期待値 → **Table 4**
  - Encode-1, 2：その他のストラテジに比べて低い
  - Encode-1, 2 vs. Encode-4～8（Post hoc に比較）
    - ✧ Encode-1, 2 は有意に低い($F(1, 105) = 1816, p < .0001$)
  - Encode-3
    - ✧ Lockout 0 では高く評価

- ✧ Lockout 400, 3200 になるに従い低くなる
  → より高いエンコードストラテジが好まれるようになる

□All-Weighted モデル

- パフォーマンス → **Fig. 2**
  - ➢ Success-Weighted に比べより人のデータにフィット
    量的，質的によく一致する（RMSE = 0.83, r^2 = 0.92，傾き = 40%一致）
- 各ストラテジの期待値 → **Table 5**
  - ➢ 最大の期待値 – 最小の期待値 → Lockout の増加に従い増加した
    - ✧ Lockout 0: 2.5
    - ✧ Lockout 400: 3.4
    - ✧ Lockout 3200: 5.5
  - ➢ Encode-1, 2 vs. Encode-4～8（Post hoc に比較）
    - ✧ Lockout 3200: Encode-1, 2 が有意に低い（$F(1, 35) = 67.7$, $p < .0001$）
      → Success-Weighted モデルと比較し，lockout 条件を通して異なるエンコードストラテジが好まれる

□Mixed-Weighted モデル

- パフォーマンス → **Fig. 2**
  - ➢ モデルの中で最も実験データにフィット
    量的，質的によく一致する（RMSE = 0.44, r^2 = 0.92，**傾き = 60%一致**）
  - ➢ Lockout 0 と Lockout 3200 のブロック数の違い
    - ✧ 人：1.84
    - ✧ **Mixed-Weighted：0.79　→ 実験データに最も近い**
    - ✧ All-Weighted：0.66
    - ✧ Success-Weighted：0.09
- 各ストラテジの期待値 → **Table 6**
  - ➢ Lockout 3200：Encode-3 ～ 6 が支配的で類似した値，それ以外は 0.5 ポイント以上低い
  - ➢ Lockout 400：Encode-3 ～ 6 に加えて，Encode-2, 8 が類似した値
  - ➢ Lockout 0：Encode-3 ～ 6 に加えて，Encode-2 が類似した値
    → Success-Weighted, All-Weighted と比較し，lockout 時間の増加は極端なエンコードストラテジを選択させなくなる

□モデル間の比較

- デフォルトの ACT-R モデル（Vanilla-Each, Vanilla-Once）
  - ➢ 簡単には人のデータとは一致しない
  - ➢ ストラテジは「成功」か「失敗」の 2 値で更新される（ひとつでもブロックを置くことができれば，「成功」となってしまう）
  - ➢ Vanilla-each モデル
    - ✧ コストによって駆動される（25～48 試行において，Encode-1 が独占）
  - ➢ Vanilla-once モデル
    - ✧ 選択されるストラテジが多少増える，しかし，Encode-1, 2 に依存

- Success-Weighted モデル
  - 成功したブロック数によってより上位の Encode ストラテジのコストが軽減される
  - コストの低減が，上位の Encode ストラテジの期待値を押し上げた
  - Encode-4～8 は非常に近い値を示す
    → Vanilla モデルよりもより上位の Encode ストラテジが選択される
    しかし，lockout 条件間で差はない
- All-Weighted モデル
  - エンコード数より少ない数しか置くことができない場合 [選択確率]が低下する（＝罰が与えられる）
  - エンコード数によって定義されたコストの削減が逆の作用を及ぼす
    → 多くのブロックをエンコードし，少ししか置くことができないストラテジが好まれる
- Mixed-Weighted モデル
  - All-Weighted モデルと同様，エンコード数より少ない数しか置くことができない場合[選択確率]が低下する
  - All-Weighted モデルとの違い
    - 実際に置くことができたブロック数によってのみコストが減少する
    - Encode-4～6 とより低い/高いストラテジが幅広く選択される
      → 質的/量的にもっとも人のデータと一致した

■An alternative ACT-R モデル

- 期待値算出以外が関係している可能性
  - 実験のデータから直接的に推定したパラメータを用いて Vanilla-Each, Vanilla-Once モデルを作成
- 結果 → **Fig. 3, Table 7, 8**
  - 擬似的 Vanilla-Each モデル
    - Vanilla-Each モデルの結果と類似
    - 常に Encode-1 を選択
    - 実験データとは一致しない（RMSE = 1.85, 傾きの一致率 = 1.4%）
  - 擬似的 Vanilla-Once モデル
    - Vanilla-Once モデルの結果と類似
    - 実験データとは一致しない（RMSE = 1.43）

    → デフォルトの期待値式では，人のデータをモデル化できない

■まとめ

- ACT-R の期待値式の制約の大きさを示した
- ACT-R への貢献
  - 「成功」/「失敗」の 2 値的判断から段階的判断への拡張（段階的判断は 2 値的判断を包含する）
  - より過激な変更が行われる前に，期待値式における現在の ACT-R の式の探求が行われるべきである
  - ACT-R の信用割り当てメカニズムの重要な限界への着目と, 信用割り当てメカニズムが調整可能か否かを議論