K-021

コードパズルによるプログラミング的思考力を考慮した理解度の推定
Estimate Programming Thinking Ability by Code Puzzle

伊東 大輝 ‡ 島川 博光 †
Hiroki Ito Hiromitsu Shimakawa

1. はじめに

プログラミング教育の現場において，採点の手軽さから知識に偏った学習方式が多い．しかし，このような学習では実際にプログラミングができるようになる学習者は多くない．このような学習者の多くは，文法の知識はあるが，与えられた問題文を実現するための知識の結びつけができていないと考えられる．指導者は，このような学習者を早期に発見し，対策を講じる必要がある．本論文では，プログラミング的思考力に着目し，学習者にコードパズルを解かせ，そのふるまいからプログラミング的思考力を考慮した理解度を推定する手法を提案する．実験の結果，80%以上の精度で理解度推定を行うことができた．実際の成績データとも 0.67 以上の相関を確認した．

2. 現在のプログラミング教育とスキーマ理論

2.1 プログラミング教育を支援する関連研究

Mysore ら [1] は，学習者が苦労している箇所を特定できる Web システム Porta を提案した．Guo ら [2] は，リアルタイムで一対多のプログラミング学習を支援するインタフェースとその実装 Codeopticon を提案した．これら多くの既存研究は，知識の側面にばかり着目しており，論理に基づく理解度推定を行っていない．Parsons ら [3] の提案した Parson’s programming puzzle は，初学者が取り組みやすいツールとして紹介されている．このツールは論理上の構成要素の習得に着目しているものの，理解度を推定するものではない．

2.2 スキーマ理論とコードパズル

認知心理学におけるスキーマは，人間の長期記憶に存在する問題解決のための考えのパターンや知識の関係性を指す．[4] 例えば，我々が‘映画を観る’という課題に取り組む時，‘支度をする’,‘駅まで行く’,‘電車に乗る’,‘映画館でチケットを買う’…のような問題解決のパターンが頭の中に存在しており，そのスキーマを今回の課題に適用する．

プログラミングにおいても，同じことが言える．この時，必要な知識と考え方のパターンがなければ問題解決の道筋をつけることはできない．すなわち，ヒトは知識を組み合わせて課題を解くと考えられる．これは，Parsons ら [3] の提案する Parson’s programming puzzle の考え方に合致している．本手法では，このスキーマ理論に基づき，コードパズルを用い，知識と論理の組み合わせからプログラミング的思考力を推定する．

3. ふるまいを用いた理解度推定

3.1 プログラミング的思考力を考慮した学習支援

本研究では，学習者がコードパズルを解くさいのふるまいから，学習者に適切なスキーマが形成されているかを判別する方法を提案する．図 1 に手法の概要を図示する．

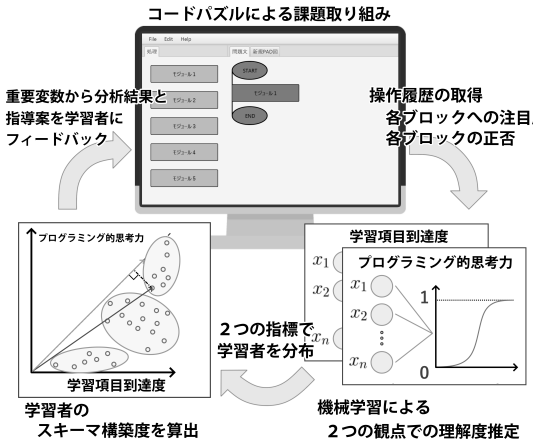

図 1: 機械学習モデルを用いたプログラミング的思考力を考慮した教育支援の概要

本手法は，学習的とその要因を判別するために，複数の学習者からコードパズルを解くさいのふるまいのデータセットを収集する．目的変数として実際の理解度は，事後のインタビューからヒトが判断する．本手法は，複数の学習者の学習行動のデータセットから，スキーマ構築度を学習項目到達度，プログラミング的思考力の両軸からロジスティック回帰・ランダムフォレスト法により分類する．本手法のモデルは変数重要度を明らかにする点から，学習者の不理解の点や解答の傾向をも明らかにすることができる．指導者は，重要度の高い変数から，スキーマ構築度に寄与したプログラミング要素を特定できる．したがって，このモデルを新規学習者に適用すると，新規学習者に足りないプログラミングの要素が特定できる．

3.2 コードパズルを用いた操作履歴の収集

本手法は，スキーマ形成有無の特定のために学習行動を収集する．学習行動の記録に用いる WEB 上で動作するオリジナルのアプリケーションのインタフェースを図に示す．

学習者は図 1 上部のように画面左側のコードブロック群から選択し，PAD[5] の形式でプログラムを記述する．解答者が見通しをもって解答しているときには，ブロックは正しい位置に迷いなく置かれる．見通しをもたない解答者は，ブロックをどこに配置するかに迷い，迷いがふるまいに表れると考えられる．本手法でのツールは，ブロックを配置するさいのふるまいを収集し，ふるまいの中の特徴を分析し理解度を推定する．

†立命館大学情報理工学部
‡立命館大学大学院理工学研究科

3.3 操作記録の取得

学習行動として大きく分けて以下を記録する．

○各ブロックの注目度

ブロックの説明を見た回数とブロックを移動させた回数を収集する．学習者は，与えられたブロックをマウスオーバーすると説明を見ることができる．学習者がブロックの説明を見た時間は，そのブロックに対する知識の有無に関連すると考えられる．あるブロックについての操作が多い学習者は，そのブロックと他のブロックの知識との組み立てができていないと考えられる．

○正否とその時間的推移

各ブロックの正否と模範プログラムとの編集距離を収集する．各ブロックには空欄が設けられていることがあり，その正否はプログラムの前後の流れを把握していなければ埋められないだろう．模範プログラムとの編集距離と近ければ，一般に論理的にブロックを組み立てられていると考えられる．

○作図に費やした時間

ツールは，問題文と作図画面が同一タブになっており，学習者はこれを切り替える必要がある．作図画面に集中できている者と課題文を読むことに必死になっている者の間には理解度に差があると考えられる．

3.4 スキーマ有無とその要因の同定

並び替え問題の学習行動からスキーマ有無の判別を行うため，ロジスティック回帰とランダムフォレスト法より，学習項目到達度に関する理解度とプログラミング的思考力に関する理解度のモデルを生成する．

学習者の理解度は，モデルの出力から，学習項目到達度とプログラミング的思考力の両軸で表した空間にプロットされる．本研究では，学習項目到達度とプログラミング的思考力の両方を持っている場合，スキーマが構築されていると同定する．このときスキーマ構築度を以下のように定義する．

$$
\begin{aligned}
f(x) &= x \\
d &= \begin{Bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{Bmatrix} \bullet \begin{Bmatrix} k \\ l \end{Bmatrix} \\
e &= (f(k) - l)^2 = (k - l)^2 \\
s &= d - e
\end{aligned}
$$

よって指導者は，これらの情報より対策を講じることができる．

4. 結果と考察

本論文で提案した手法の有効性を検証するために，実験を行った．被験者はプログラミングを学習し始めた学部 1 回生からプログラミングに慣れている大学院 1 回生までを含む 17 名である．被験者は課題 1 に関して最短で 7 分，最長で 40 分かかり，到達度はさまざまであった．

表 1: 学習項目到達度分類モデルの性能

LR	
正解率	0.82
精度	0.78
再現率	0.88
F 値	0.82

RF	
正解率	0.82
精度	0.86
再現率	0.75
F 値	0.80

表 2: プログラミング的思考力分類モデルの性能

LR	
正解率	0.82
精度	0.79
再現率	1.00
F 値	0.88

RF	
正解率	0.94
精度	0.92
再現率	1.00
F 値	0.96

表 1, 2 に本実験により作成したモデルの正解率を示す．どのモデルでも正解率は 8 割を超えており，ある程度精度の高いモデルであると言える．また 3.4 節で記述したようにスキーマ構築度を計算すると，実際の成績との相関は 0.67 になった．本実験の結果は，コードパズル解答時の学習者の操作履歴から，プログラミング的思考力を考慮した理解度の推定が可能であることを示している．本手法で推定されるスキーマ構築度は，十数サンプルにラベル付けを行いモデルを作れば，一度に多くの学習者の理解度を測定することができる．特にロジスティック回帰で出力される分類確率は 0.0 から 1.0 までの連続値であることから，閾値を適切に設定することによって，独力で解かせるべき学習者と，要指導の学習者を切り分けることができる．

5. おわりに

本論文では，プログラミング教育の現場で学習者のプログラミング的思考力を考慮した理解度を推定する手法を提案した．本手法は，コードパズル課題の解答過程から，学習者の理解度とその要因を明らかにする．指導者は理解に躓いている学習者を早期に発見することができ，より適切な対策を講じることができる．今後はシステム構築にかかるコストを削減するため，手法を改良する．

参考文献

[1] Alok Mysore and Philip J Guo. Porta: Profiling software tutorials using operating-system-wide activity tracing. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pp. 201–212, 2018.

[2] Philip J Guo. Codeopticon: Real-time, one-to-many human tutoring for computer programming. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 599–608, 2015.

[3] Dale Parsons and Patricia Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pp. 157–163, 2006.

[4] Wolfgang Schnotz and Christian Kürschner. A reconsideration of cognitive load theory. *Educational psychology review*, Vol. 19, No. 4, pp. 469–508, 2007.

[5] 二村良彦, 川合敏雄, 堀越彌, 堤正義ほか. Pad (problem analysis diagram) によるプログラムの設計および作成. 情報処理学会論文誌, Vol. 21, No. 4, pp. 259–267, 1980.