N-002

ソースコードに基づいた特徴量作成による プログラミング課題での見通しの評価

Evaluating Prospects in Programming Assignments by Creating Features Based on Source Code

小林 智晴† 原田 史子† 島川 博光†
Tomoharu Kobayashi Fumiko Harada HIromitsu Shimakawa

1. はじめに

日本では 2020 年度から 2022 年度にかけて、小・中・高の学校教育でのプログラミング教育の必修化がなされた。導入に伴い、文部科学省が公表しているプログラミングの手引き[1]では、プログラミング的思考を育むことを重要視している。この思考能力を育成するうえで課題となるのは、集団学習における評価法である。プログラミングの指導であれば、プログラミング中の実践的な取り組み過程を観察したうえで評価することが好ましい。しかし、指導者が生徒の思考を推察するには、ある程度の時間その取り組みを観察するほかない。加えて、集団学習において、複数の生徒の思考を同時に観察・評価することは困難である。よって、プログラミング教育における円滑な指導のため、生徒らの取り組み過程を包括的に観察し、思考能力を評価する手法が求められる。本論文では、プログラミング学習者の思考を機械的に評価するうえで、解答過程のソースコードに基づく機械学習向けの特徴量を作成する手法を提案する。

2. 関連研究

2.1 日本のプログラミング教育

プログラミング教育で育む資質・能力のうちの１つに、“思考力、判断力、表現力等”がある[1]。プログラミング的思考を育成することは、新たな学習指導要領における重要な要素である。しかし、この思考能力についての明確な指導評価法は存在せず、その指導評価は、指導者の実地的な判断に頼るところが大きいと思われる。よって、本論文ではプログラミング的思考の評価基準となる定性的な尺度の役割を持つ特徴量および、その作成手法を提案する。

室伏[2]の調査によると、小学校プログラミング教育の報告レポートの中での実施形態は、Scratch や Viscuit といったビジュアル言語のプログラミングツールが過半数を占めている。これらは操作が感覚的で初学者にも扱いやすく、文部科学省の提示する研修教材にも、Scratch を利用した多角形を描く課題が示されている。また、谷川ら[3]の研究では解答過程の記録から学生の習得項目を断定したほか、学生間で間違い方に同様のパターンが存在することを示した。これは、単一の課題中では学生の思考パターンが識別可能な程度に分類できることを示唆している。このことから、図形描画課題での思考パターンを判別することは、初学者に対するプログラミング教育において思考能力に寄り添った指導評価を確立する要因になると考えられる。よって、本論文では図形描画のプログラミングにおける取り組みを対象として、それぞれの学習者の各時点での思考パターンを適切に判別することを目標とする。また、時系列的な関係を表現することで、学習者の課題中での見通しの評価への発展も考える。

2.2 計算論的思考

プログラミング的思考の概念は、計算論的思考と訳される”Comptational Thinking”[4]がもとになっている。海外では計算論的思考に関する取り組みが盛んに行われている。Tikva ら[5]や Fagerlund ら[6]は、計算論的思考に関する文献調査の結果を示している。それらの結果によると、計算論的思考の評価は調査研究の中でも関心が強い。また、プログラミング的思考も同様であるが、思考法の実態をつかむこと自体が難しい点から、計算論的思考に関する指導基準を策定するため、評価法に関する多くの研究がなされている。近年の事例として、Guggemos ら[7]は高校生を対象とした測定テストにより、生徒の計算論的思考と周辺要因との関連性を示した。Wei ら[8]は小学校科目での対照実験を行い、計算論的思考能力の向上に有効なアプローチの方向性を示した。これらの研究はプログラミング教育における評価尺度の確立を手助けするものである。しかし、生徒の特性が地域性に依存するという懸念や、特定のカリキュラムに基づいていることから、日本のプログラミング教育においても有効である保証はない。このような考えを受けて、本論文では環境や個人に依存しないソースコードでの解答に注目することで、幅広く実践可能な評価法を実現したいと考える。

3. ソースコードからなる特徴量ベクトルの作成

3.1 ソースコードの特徴量化

本章では、図形描画のプログラミングにおける解答過程のソースコードをもとに、機械的な判別を実現するための特徴量の作成手法を示す。手法の概要を図１に示す。図１の手順は主に３つに分かれている。以下その流れに沿って手法を紹介する。

はじめに、プログラミングの解答過程を時系列データとして扱うべく、課題中のソースコードを逐次的に記録しておく。ソースコードは学習者が構築しようとする考えそのものである。2.1 節での谷川ら[3]の研究では、ソースコードを実行したさいの関数呼び出しの記録を利用している。このことから、ある時点での学習者の到達度合いをソースコードから断定することは可能であろう。また、課題の表現が一意的であれば、学習者は一意に決まった順序で部分的項目を満たし、段階的に正答に近づいていくと考えられる。この場合、それぞれの学習者がどの程度課題を進めら

†立命館大学 Ritsumeikan University

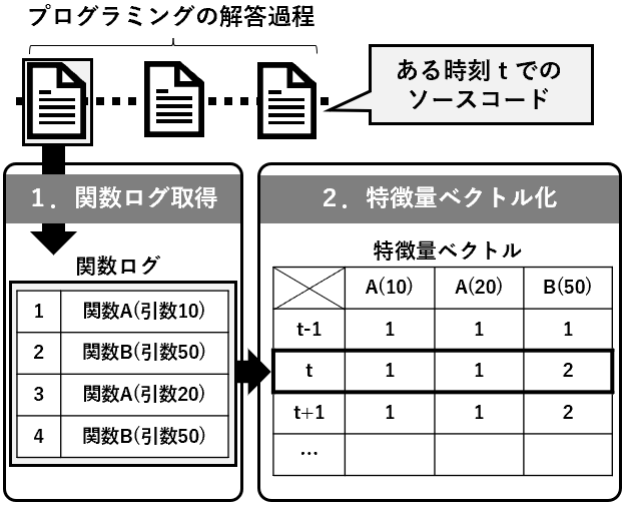

図 1　ソースコードの特徴量変換

れているかの到達度を、ソースコードから判別することができるといえる。さらに、解答過程の時系列性を保つことで、課題の進展をもとに学習者の見通しについても判別できるだろう。こういった考えのもと、本論文ではソースコードを学習者の思考ないしは見通しを評価するためのデータとして扱う。

次に、関数ログを取得する。各時点のソースコードを実行したさいの関数呼び出し記録を羅列した形で記録する。このとき、ソースコードひとつに対して関数ログもひとつ生じるため、解答過程の時系列長と同じだけの数の関数ログが生成されることとなる。関数ログを利用することで、テキストベースでは表現し難いであろう繰り返しや条件分岐で実装された箇所についても、描画に則した順序関係を表現できる。

最後に、取得した関数ログを特徴量ベクトルに変換する。先で取得した関数ログ群は、それぞれの時点でのソースコードを参照しているため、各時点でのデータ長が異なる。それらを形式の揃った時系列データとして扱えるよう、ある時点 t での関数ログを 1 行分のデータとして、単一の時系列ベクトルへ集約する。図 1 左側の関数ログに関数 A・B の任意の引数を伴う呼び出しが 4 つ記録されている。そのため図 1 右側には 4 回分のログについて、ある関数がある引数で呼び出された回数という列の振り分けで、1 行分のベクトルとして変換されている。

このような手順で、ソースコードの関数呼び出し記録に基づいた特徴量を作成することができる。次節では、この特徴量の特性について議論する。

3.2 特徴量ベクトルの特性

図 1 の手順で作成した特徴量ベクトルの特性について、留意すべき点を述べる。各列は関数と引数の組み合わせによって構成される。そのため、課題ごとに必要となる列の集合は異なる。また、課題の正答が一意的であれば、正答のベクトル表現に必要な列の組み合わせも一意に定まる。ただし、解答過程では不適切な引数で関数が呼び出される場合もある。これに関しては、課題を表現するのに不適切な引数での呼び出し回数を、その他の呼び出し回数の列として表現するべきだと考えられる。

さらに、呼び出し回数のみのベクトルでは、関数ログ上の順序を反映できない。そのため、実際には関数ログから特徴量ベクトルに変換する過程で、さらなる特徴量エンジニアリングを行うべきである。具体的には、”A→B”, ” B→A”, ”A →C”…といった特定の順序での関数呼び出し回数を抽出する。これにより、関数ログの順序関係を特徴量に落とし込み、思考パターンとの関係を適切に表現できる。

3.3 検証における考察

前節までの手法案を実証するため、10 名の大学生に対して図形描画課題実験を実施した。課題の内容については紙面の都合上割愛するが、繰り返しや条件分岐を含んだ内容である。3.1 節に示した手法のとおりソースコードから特徴量ベクトルを作成した。そのさい、3.2 節に示したように、例外的な引数、関数の組み合わせを含む特徴量とした。

この特徴量を観察した結果、３つの事象を確認した。１つ目に、正答者と正答例の特徴量表現は一意的に一致した。２つ目に、数人の被験者間で途中段階の解答パターンや間違いパターンにおける特徴量が、一致もしくは類似する結果となった。3 つ目に、課題に示す内容と異なる独自の方針で描画を行った被験者について、特徴量ベクトルが想定されるものから逸脱する値となった。これらのことから、提案する特徴量は課題中の思考パターンを十分に表現しており、機械的な判別も可能であると考えられる。

4. おわりに

本論文では、図形描画のプログラミング中の解答過程のソースコードに基づき、学習者の見通しを評価するための特徴量を作成する手法を提案した。また、その特性について検証結果を交えて考察した。この特徴量について、今後の目標はプログラミング教育における評価尺度としての役割を確立することである。そのために、特徴量についての分析を続けるとともに、新たに課題実験によりデータを収集する。また、時系列性を踏まえたうえで機械学習モデルによる判別法を考案・検証し、その効果と指導評価への応用方針を議論する。

参考文献

[1] 文部科学省,「小学校プログラミング教育の手引」（第三版）＜ https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf＞（アクセス日：2022 年 6 月 17 日）
[2] 室伏春樹. 小学校プログラミング教育の現状分析と課題. 静岡大学教育実践総合センター紀要. Vol. 32, p. 119-126. (2022)
[3] Kohei Tanigawa, Fumiko Harada, Hiromitsu Shimakawa, “Detecting Learning Patterns during Exercise from Function Call Logs”, International Journal of Advanced Computer Science, Vol.1, No.1, pp.30-35, (Jul. 2011)
[4] Wing, Jeannette M. "Computational thinking." Communications of the ACM 49.3 (2006): 33-35.
[5] Tikva, Christina, and Efthimios Tambouris. "Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review." Computers & Education 162 (2021): 104083.
[6] Fagerlund, Janne, et al. "Computational thinking in programming with Scratch in primary schools: A systematic review." Computer Applications in Engineering Education 29.1 (2021): 12-28.
[7] Guggemos, Josef. "On the predictors of computational thinking and its growth at the high-school level." Computers & Education 161 (2021): 104060.
[8] Wei, Xuefeng, et al. "The effectiveness of partial pair programming on elementary school students’ computational thinking skills and self-efficacy." Computers & education 160 (2021): 104023.