プログラミング学習ログパターンに基づく行動変化分析手法の提案

A Method for Analyzing Behavioral Changes Based on Programming Learning Log Patterns

小舟　康予 † 小比賀　亮仁 ‡ 小板　隆浩 ‡ 嶋利　一真 † 松本　健一 †
Yasuyo Kofune Akihito Kohiga Takahiro Koita Kazumasa Shimari Kenichi Matsumoto

奈良先端科学技術大学院大学 † 同志社大学 ‡
Nara Institute of Science and Technology Doshisha University

1. はじめに

2020 年度の学習指導要領の改訂により，小学校においてプログラミング教育が必修化されるなど，初等中等教育段階におけるプログラミング教育の普及が急速に進んでいる（文部科学省, 2017）．また，中央教育審議会（2021）では，すべての子どもたちの可能性を引き出すために，「個別最適な学び」と「協働的な学び」の一体的な充実が重要であると提言されている．これに伴い，学習者一人ひとりの理解度や課題に応じた柔軟な学習支援の必要性が高まっている．

プログラミング教育の現場においても，学習者の理解の進み具合やつまずきを把握し，適切なタイミングで支援を行うことが学習効果の向上に有効であるとされている．しかし，現在，多くの授業や学習支援システムでは，最終提出物の正誤や完成度といった「結果」に基づく評価が中心であり，「どのように学んだか」といった学習過程を捉える仕組みは十分に整備されていないのが実情である．

このような課題に対し，近年ではプログラミング学習ログを活用し，学習プロセスを可視化する取り組みが進んでいる．例えば，Altadmri ら（2015）は大規模なコンパイルログを用いてエラーの頻度や修正時間を分析し，初学者が陥りやすいつまずきを体系的に整理した [1]．また，Estey ら（2017）は，プログラミング学習ログの時系列的な変化（trajectory）を分析することで，「一時的なつまずき」と「持続的なつまずき」を識別し，支援対象となる学生をより的確に抽出できることを示している [2]．さらに，Gao ら（2021）は，初期課題における行動パターンが最終成績と強く相関することを示し，プログラミング学習ログによるパフォーマンス予測の可能性を提示した [3]．

これらの研究はいずれも，プログラミング学習ログを通じて学習者の傾向や状態を把握できる可能性を示している．一方で，学習者の行動が時間の経過とともにどのように変化するか，またその変化をどのように定義し，構造的に分類・抽出するかといった観点については，十分に体系化されているとは言いがたい．特に，エラー数やヒント利用頻度といった定量指標の値を断片的に評価する手法は多く存在するが，「編集が安定してきた」「同じ箇所を何度も繰り返し修正している」「コードの入力が停滞している」といった行動変化パターンそのものを明示的に捉えようとする試みは限られている．

本研究では，プログラミング学習者のコード編集・実行ログにおける行動変化を分析し，その変化の特徴に基づいて学習状態を「安定」「反復」「停滞」に分類するルールベース手法を提案する．この手法により，学習過程における理解の進行やつまずきの兆候を可視化し，個別支援への応用が期待される．

2. 先行研究

本章では，関連する先行研究を取り上げ，本研究との関連を整理する．

2.1 プログラミング学習ログの可視化

Blikstein（2011）は，NetLogo を用いた学習環境で収集したコード編集ログをもとに，学習者のコードスナップショットを時系列に可視化し，学習の進め方を複数のスタイルに分類した [4]．この研究は，ログ可視化により行動軌跡の多様性を捉えた点に意義がある．一方で，分類に用いる特徴量が明示されておらず，分類ルールの構造も定義されていないため，汎用的な状態推定手法としての応用は難しい．Charitsis ら（2022）も，Java プログラミング課題における編集スナップショットを可視化するツールを開発し，学習者の問題解決過程の把握と教員によるフィードバック支援を目的とした [5]．本ツールは視覚的支援環境として有効であるが，学習状態の分類やその変化を対象とした分析は行っておらず，学習者の理解状態を構造的に推定する枠組みとは異なる．

2.2 エラー傾向と行き詰まりの検出

Altadmri ら（2015）は，世界中の学習者から収集した 3,700 万件以上の Java コンパイルログを分析し，エラー頻度・修正時間・発生タイミングといった定量情報を網羅的に可視化した [1]．この研究は，初学者が直面しやすいエラーの傾向を把握する上で有用である．しかしながら，分類は「エラーの種類」に留まり，学習者の行動変化や理解状態の推定には踏み込んでいない．藤原ら（2018）は，プログラミング演習時に収集したソースコードのスナップショットを分析する手法を提案した [6]．この手法では，各スナップショットと最終提出物との差分（編集距離）を

† 奈良先端科学技術大学院大学,8916-5 Takayama-cho, Ikoma, Nara 630-0192, JAPAN
‡ 同志社大学,1-3 Tataramiyakodani, Kyotanabe, Kyoto 610–0394, Japan

もとに「推定残作業量」を定量化し，その量が停滞あるいは増加する区間を「行き詰まり」として検出する．提出後の事後的な分析を前提としており，リアルタイムでの検出や即時支援には限界がある．一方で，検出された行き詰まりは「言語仕様の理解不足」や「アルゴリズムの理解不足」などの要因に分類され，講師によるつまずきの理解と的確なフィードバックに資する点で意義がある．Kohn（2019）は，Python の構文エラーにおいて，エラーメッセージが指す内容と実際の誤り原因との間にずれが生じることに着目し，その性質を分析することで自動検出の限界を指摘した [7]．高校生の Python プログラム約 4,000 件を分析した結果，エラーの約 30 %は綴り間違いや括弧の欠落といった単純なミスに起因していたが，複雑な誤りの正確な診断にはプログラム全体のゴールやプランといった設計意図の把握が必要であり，従来の構文解析ベースの手法では対応が困難であることが示された．

これらの研究は，いずれも学習者のエラーや行き詰まりに関する有益な知見を提供しているが，主に事後的な分析に依存しており，学習者の行動変化をリアルタイムかつ構造的に捉える枠組みは提示されていない．また，学習状態を明示的に定義・分類するアプローチも見られず，汎用的かつ即時的な学習支援への展開には課題が残る．

2.3　行動変化と理解度の推定

プログラミング学習者の行動ログを分析し，学習状態を把握する研究は複数存在する．Estey ら（2017）は「一時的なつまずき」と「持続的な困難」を区別するために，プログラミング行動の変化量を測定する「trajectory metric」を提案した [2]．これにより，単一時点の評価では見過ごされる学習の進展や停滞を捉える枠組みを提示している．同様に，槇原ら（2018）は，修正・コンパイル・実行を繰り返す「探索的プログラミング行動」を検出することで，学習者のつまずきをリアルタイムに特定する手法を提案している [8]. Gao ら（2021）は，初期課題における「編集→実行」といった行動シーケンスの頻出パターンを抽出し，その後のコース成績を約 79%の精度で予測可能であることを示した [3]．抽出されたパターンは「抽象化」や「デバッグ」といった具体的なプログラミング行動と関連づけられ，解釈可能な示唆を与えるものだが，学習者の状態を「高成績者／低成績者」の二値で静的に分類するため，学習過程における動的な状態変化の追跡には至っていない．

これらの研究は，それぞれ異なるアプローチで学習者の行動変化や状態の検出を試みており，個々に重要な知見を提供している．しかしながら，「探索」「一時的なつまずき」「持続的な困難」といった個別の状態は定義されているものの，それらの状態がどのように遷移し，相互に影響し合うのかを構造的にモデル化する統一的な枠組みはまだ提示されていない．そのため，学習者の状態を多角的に捉え，その変化に応じて即時的かつ体系的なフィードバックを行う，という次のステップへの展開には依然として課題が残ると言える．

2.4　本研究の立ち位置と先行研究の整理

これまでに紹介した先行研究と本研究の比較を表 1 に示す．プログラミング学習ログの可視化，エラー検出，行動変化の各アプローチはそれぞれに有益な知見を提供しているが，いずれも学習者の状態を構造的かつ明示的に分類し，リアルタイムに可視化・支援へとつなげる枠組みは十分に整備されていない．本研究は，各種特徴量の変動パターンに着目したルールベース手法を設計することで，分類の明示性と即時性を両立し，汎用的な支援システムへの応用を可能にする点に特徴がある．

このように，本研究は従来の可視化・予測・検出に加えて，状態の分類・可視化・即時的支援という多段階の学習支援フローを視野に入れた新たな枠組みを提供する．

本研究では，以下の点で新たな貢献を目指す．

- 「安定」「反復」「停滞」といった行動変化の状態を明示的かつ構造的に定義
- 編集頻度・行数変化・実行回数・実行成功率・AST 類似度といった特徴量に基づくルールベースの分類手法
- リアルタイム推定・軽量実装を想定した分類ルール設計
- ダミーデータによる初期的検証と今後の実データ展開を視野に入れた設計

3.　研究目的とアプローチ

3.1　研究の背景と課題

プログラミング学習者の理解状態やつまずきの兆候を把握するためには，最終成果物だけでなく，学習過程における行動の変化を捉えることが重要である．近年では，コードの編集・実行履歴などの学習ログを用いて，学習プロセスを可視化・分析する研究が進められている（第 2 章参照）．しかしながら，学習者の行動を構造的に分類し，時系列的な状態変化として明示的に把握する枠組みは，まだ十分に確立されていない．特に，「安定して作業を進めている状態」と「つまずきや試行錯誤が続く状態」を区別するための定量的で明示的な分類基準や，それをリアルタイムかつ視覚的に把握可能な手法には課題が残っている．また，実運用を想定する際には，複雑な機械学習モデルではなく，軽量かつ解釈可能なルールベースのアプローチが求められる．

本研究では，以下の課題意識に基づき，プログラミング学習ログの分析手法を設計する．

- 学習者の行動変化（編集頻度の変化，繰り返し編集，エラー修正の停滞など）を時系列的に捉える必要がある

表 1: 先行研究との比較

	【2.1 学習ログの可視化】 Blikstein (2011), Charitsis (2022)	【2.2 エラー検出】 Altadmri (2015), 藤原 (2018), Kohn (2019)	【2.3 行動変化】 Estey (2017), 槇原 (2018) , Gao (2021)	本研究
対象	編集スナップショット	エラー履歴・編集差分	行動シーケンス・変化量	編集・実行・構造変化
分析粒度	静的スナップショット	静的／事後分析	行動変化（時系列）	特徴量変化（時系列）
状態分類	×（分類なし）	△（一部分類）	○（探索・困難など）	◎（安定・反復・停滞）
リアルタイム性	×	×	△（一部）	◎
応用性	教師支援・可視化	エラー理解・事後支援	状態推定・予測支援	状態推定・可視化・自動支援

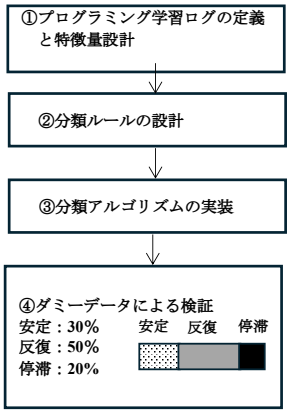

図 1: 行動変化分析に基づく学習状態分類の全体構成

- 学習状態を分類・可視化するための，簡潔で解釈可能な分類基準が必要である

3.2 研究目的

本研究では，プログラミング中の編集・実行履歴およびコード構造の変化に着目し，これらの時間的な変動パターンに基づいて，学習状態を「安定」「反復」「停滞」に分類する軽量な手法を提案する．編集頻度，行数変化，実行回数，実行成功率，AST 類似度といった複数の特徴量を時系列的に捉え，行動の変化を定量的に把握・分類することで，学習プロセスの理解と可視化を可能にする．従来の研究が単一時点の行動や静的なスナップショットに依存していたのに対し，本研究では特徴量の変化に基づく明示的な分類基準を設計し，リアルタイムでの状態可視化や自動フィードバックへの応用を視野に入れた汎用的な枠組みの構築を目的とする．

3.3 研究アプローチ

行動変化分析に基づく学習状態分類の全体構成を図 1 に示す．具体的には，編集頻度，行数変化，実行回数，実行成功率，AST 類似度を抽出，特徴量の統計的分布に基づきしきい値を用いた分類基準を設計，ウィンドウ単位で学習状態を推定，各分類状態の妥当性を評価する．

このような構造的かつ軽量な分類手法により，将来的にはリアルタイムな学習支援環境への応用も期待される．

4. 分類手法の設計

本研究では，プログラミング学習ログをもとに，学習状態を「安定」「反復」「停滞」の 3 つに分類するためのルールベース手法を設計する．対象とするログは，Visual Studio Code 上で取得される編集（edit）・実行（run）・保存（save）である．

4.1 特徴量の定義と抽出

学習状態を分類するため，以下の 5 つの定量的特徴量を 30 秒ウィンドウ単位で集計・抽出する．抽出した 5 つの特徴量を表 2 に示す．

edit_count_sum は，VS Code の onDidChangeTextDocument イベントの発火回数を基準に算出されたものである．これは，ユーザによる文字単位の入力・削除といった操作をすべて個別にカウントしたものであり，必ずしも意味単位の編集とは一致しない．今後，編集間隔や構文変化に基づく意味単位での集約も課題として残される．

lines_changed_sum は，各ウィンドウ内においてユーザが編集したコードのうち，変更・追加・削除された行数の合計である．VS Code における onDidChangeTextDocument イベントから取得される range 情報に基づき，編集が行われた範囲の行数をカウントして集計している．

これらの特徴量は，学習状態の推定に有効な基礎指標となる．いずれも逐次的な集計が可能な軽量かつ解釈可能な指標であり，リアルタイム処理や可視化への応用を考慮して設計している．

4.2 状態の定義と分類ルール

プログラミング学習中に観察される編集・実行といった行動の変化に着目し，行動パターンを 3 つの学習状態に分類する．

本研究では，以下のように状態を定義し，行動変化の特徴量に基づくルールを設計する．

- 安定：編集・実行の頻度が少なく，成功率が高く，構造変化も小さい状態．理解が進み効率的に作業できていると推察される
- 反復：同一箇所を繰り返し編集・実行している状態．同一箇所の試行錯誤が続いていると考えられる
- 停滞：編集や実行がほとんど行われず，新たな試行が

表 2: 各ウィンドウに対して抽出される特徴量

特徴量名	説明	取得ログ
edit_count_sum	編集の合計回数	edit
lines_changed_sum	編集による変更行数	edit
run_count	実行回数	run
success_rate	実行における成功率	run
ast_sim_avg	構造変化の類似度	edit，save

なされていない状態

学習状態の分類にはスコアベースの手法（以下，スコアベース分類手法）を採用する．各特徴量について，状態ごとに条件を定め，それを満たすたびにスコアを+1 する．最もスコアの高い状態をそのウィンドウのラベルとする（同点時は「停滞＞反復＞安定」の優先順位で分類，すべて 0 なら「未分類」）．状態ごとのスコア加算の条件を表 3 に示す．

特徴量スコアに基づく学習状態分類の疑似コードをリスト 1 に示す．

リスト 1: 学習状態分類の疑似コード

```
for each window in data:
    scores = {"安定": 0, "反復": 0, "停滞": 0}

    # 特徴量ごとに条件を判定し，スコアを加算
    if editCount_sum <= 2: scores["安定"] += 1
    if editCount_sum >= 3: scores["反復"] += 1
    if editCount_sum <= 1: scores["停滞"] += 1

    if linesChanged_sum <= 3: scores["安
        定"] += 1
    if linesChanged_sum <= 5: scores["反
        復"] += 1
    if linesChanged_sum <= 1: scores["停
        滞"] += 1

    if runCount >= 1: scores["安定"] += 1
    if runCount >= 3: scores["反復"] += 1
    if runCount <= 1: scores["停滞"] += 1

    if successRate >= 0.80: scores["安
        定"] += 1
    if successRate <= 0.70: scores["停
        滞"] += 1

    if astSimAvg >= 0.85: scores["安定"] += 1
    if astSimAvg >= 0.80: scores["反復"] += 1
    if astSimAvg >= 0.85: scores["停滞"] += 1

    # 最もスコアの高い状態を分類結果とする
    if max(scores.values()) > 0:
        label = argmax(scores)
    else:
        label = "未分類"

    assign label to window
```

スコアベース分類手法により，すべての条件を満たさなくても分類が可能となり，プログラミング学習ログに含まれる揺らぎや欠損を許容できる柔軟な判定が実現される．

4.3 ルール設計の方針としきい値の根拠

学習状態を分類するルールの設計は，以下の手順に基づいて行った．

(1) 教育実践や文献に基づく行動傾向の仮説化
　例：「安定」では編集頻度・構造変化が少なく成功率が高い．

(2) ダミーデータの統計的分布（四分位値など）との対応付け
　例：edit_count_sum $\leqq$ 2 は第 1 四分位に相当し，活動が収束している状態を示す．

このようなルール設計により，従来のようにブラックボックス的なモデルに頼らず，特徴量の意味に基づいて学習状態を解釈・説明することが可能となる．第 5 章にてダミーデータを用いて初期的な妥当性を検証し，今後の実データ適用に向けた再調整を想定している．

4.4 実装上の工夫と応用可能性

本手法は，軽量かつ実用的な分類を実現するため，以下の点に配慮して設計している．

- ウィンドウ幅の設計：30 秒というウィンドウ幅は Blikstein（2011）[4] などの研究とダミーログの観察結果（10〜30 秒）に基づいて設定したものである．学習状態の変化を捉えるのに十分な粒度を確保しつつ，複数イベントを 1 単位にまとめて安定した判定ができる．
- 不完全ログへの耐性：複数の特徴量によるスコア加算方式により，一部のログが欠損していても他の情報から補完的に分類可能である．
- AST 類似度の扱い：高精度な構文比較は計算コストが高いため，構造変化の近似や局所変化量を使った簡易指標として導入する（今後さらなる最適化を検討）．
- 分類結果の可視化：ウィンドウごとの状態をタイムラインとして表示することで，学習者の状態遷移やパターンを視覚的に把握できる．

上記のような工夫により，本手法はリアルタイム支援

表 3: 学習状態分類のためのスコア加算ルール
※各条件を満たすたびに該当状態のスコアが+1 加算される．

特徴量名	安定	反復	停滞
edit_count_sum	$\leqq 2$	$\geqq 3$	$\leqq 1$
lines_changed_sum	$\leqq 3$	$\leqq 5$	$\leqq 1$
run_count	$\geqq 1$	$\geqq 3$	$\leqq 1$
success_rate	$\geqq 0.80$	-	$\leqq 0.70$
ast_sim_avg	$\geqq 0.85$	$\geqq 0.80$	$\geqq 0.85$

表 4: 各ウィンドウの状態ラベル付与に用いた確率分布

状態	割合 [%]
安定	30.0
反復	50.0
停滞	20.0

や学習ダッシュボードへの応用も見据えた設計となっている．

5. 分類結果の評価

本章では，第 4 章で提案したスコアベース分類手法の有効性を検証する．ダミーデータの生成方法と前提条件を述べた後，分類結果の可視化，出現割合，正解ラベルとの一致度に基づく評価を行い，最後に誤分類傾向と今後の改善点を考察する．

5.1 ダミーデータの概要と生成条件

評価には，実データ収集が未完了であることを踏まえ，提案する分類ロジックの基本的な妥当性と実現可能性を検証する第一歩として，学習行動を模したダミーデータを用いて初期評価を行った．ダミーデータは，合計 100 個の 30 秒ウィンドウから構成されており，プログラミング課題に取り組む初学者の典型的なセッション（例：50 分程度）を模したものとなっている．30 秒ごとのウィンドウに対して以下の 5 つの特徴量を割り当てた．

- edit_count_sum（編集回数）
- lines_changed_sum（変更行数）
- run_count（実行回数）
- success_rate（実行成功率）
- ast_sim_avg（AST 類似度）

各ウィンドウに「安定」「反復」「停滞」のいずれかの状態ラベル（以下，ground truth）を付与した．状態ラベルの付与割合を表 4 に示す．

5.2 分類結果の可視化と一致度評価

スコアベース分類手法を適用した結果，ウィンドウごとの学習状態はタイムライン上に可視化され，時間的な学習変化が視覚的に把握可能となった．その結果を図 2 に示す．また，分類された状態の出現割合を表 5 に示す．

出現比率が極端に偏ることなく，3 つの状態がバランスよく分類されていることが確認できた．分類精度は，ground truth との一致率を Accuracy および Macro-F1 スコアにより評価した．その結果を表 6 に示す．

表 5: 分類結果の出現割合

状態	出現割合 [%]
安定	31.0
反復	36.0
停滞	33.0
未分類	0.0

表 6: 分類結果の正確性の評価

状態	精度評価 [%]
Accuracy	66.0
Macro-F1	66.5

5.3 誤分類の分析

ground truth と分類結果の関係を表した混同行列を図 3 に示す．混同行列をもとに，本研究のスコアベース分類手法による各状態の分類精度と誤分類傾向は，以下のように整理できる．

- 「安定」状態（正解: 31 件）のうち，27 件が正しく分類されており，高い識別精度を示した．一方で，1 件が「反復」，3 件が「停滞」と誤分類されており，特徴量の一部（例：低い編集頻度）により混同が生じた可能性がある．
- 「反復」状態（正解: 36 件）は 21 件が正しく分類されたが，14 件が「安定」，1 件が「停滞」に分類された．この誤分類は，編集や実行の頻度がやや少ない「反復」の活動が，「安定」と見なされたことによると考えられる．
- 「停滞」状態（正解: 33 件）は 18 件が正しく分類されたが，14 件が「安定」，1 件が「反復」と判断された．成功率が高く，コード構造も安定していた場合，操作の停止にもかかわらず「安定」と誤認された可能性がある．

この結果から，短期的な静止状態（反復や停滞の初期段階）は「安定」と誤認しやすい．現行ルールでは時間的な連続性（前後の文脈）を考慮していないため，瞬間的な特徴量で判断される．実行成功率や AST 類似度など，一部

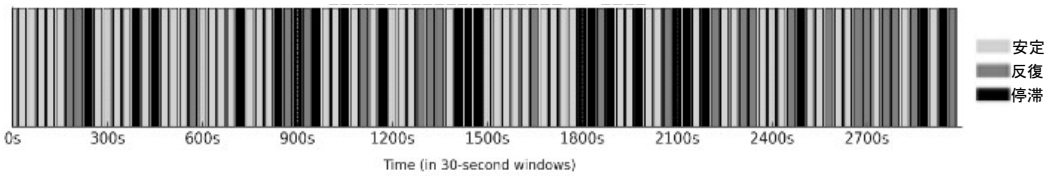

図 2: 分類結果のタイムライン（スコアベース分類）

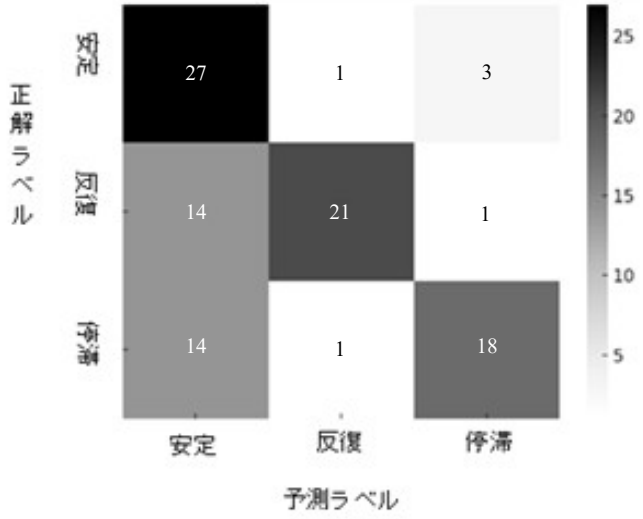

図 3: スコアベース分類と正解ラベルの比較

の特徴量が「安定」に強く引っ張る傾向がある．「停滞」は動きがないため，特徴量として値が出にくく，実行成功率や AST 類似度などの特徴量が優先されてしまう傾向がある．今後は，より繊細な特徴量の設計や，時間的変化を加味した文脈的判断の導入が，さらなる分類精度の向上に寄与すると考えられる．

5.4 限界と今後の改善点

本研究はダミーデータに基づく初期的な検証であり，実データに対する汎用性・精度の検証は今後の課題である．また，現行の分類手法には以下の改善点が挙げられる．

- 時間的遷移を考慮した連続的分類（例：HMM や状態遷移モデル）の導入
- データ分布に応じた動的なしきい値の最適化
- 一部特徴量が欠損した場合の補完・ロバスト性の向上
- 分類結果に基づいた支援タイミング・フィードバック方針との接続

特に，「誤分類されやすい一時的な停滞や反復の兆候」を見逃さないためには，今後の実装において前後ウィンドウの遷移パターンを踏まえた分類強化が鍵となる．

6. まとめと展望

本研究では，プログラミング学習ログに基づき，学習状態を「安定」「反復」「停滞」の 3 つに分類するルールベース手法を提案し，ダミーデータを用いてその妥当性を初期的に検証した．ここでは本研究の成果を総括し，今後の課題と応用可能性について展望する．

6.1 本研究の成果

本研究の主な成果は以下の 3 点に集約される．

- プログラミング学習ログにおける行動変化の分析を通じた学習状態（安定・反復・停滞）の定義と分類ルールの提示
  編集頻度，実行回数，成功率，AST 類似度といった解釈可能な特徴量を用いて，状態をルールベースで推定可能とした．
- 軽量かつリアルタイム適用可能な分類アルゴリズムの実装
  スコアベース分類手法により，柔軟性と計算効率を両立し，実環境での活用に向けた基盤を構築した．
- ダミーデータによる初期的な有効性の検証
  Accuracy 66%，Macro-F1 66.5% の分類精度を達成し，タイムライン上で学習状態の変化を可視化する可能性を示した．

6.2 今後の課題と発展の方向

今後は，以下の 4 つの観点からの発展が求められる．

- 実データによる精度検証とルール最適化
  現在の評価はダミーデータに依存しており，今後は実際の学習者ログを用いて，分類ルールの妥当性・精度を再検証する必要がある．特に，特徴量のしきい値設定や分類誤差の傾向分析を通じて，ルールの最適化を進める．
- 時系列変化を踏まえた動的推定モデルへの拡張
  学習状態は連続的に遷移するものであるため，ウィンドウ間の関係性を取り入れたモデル（例：HMM，状態遷移ネットワーク）によって，より文脈的な分類が可能になると期待される．
- 教育支援への応用とパーソナライズ化
  「反復」や「停滞」状態をリアルタイムに検出し，学習者に適切なヒントや支援を提供するインタラクティブな支援システムへの応用が重要である．また，学習者の行動傾向や課題特性に応じて，分類ルールを個別最適化するパーソナライズ手法の導入も重要な検討課題である．
- 状態定義の拡張と未分類状態への対応
  本研究では，「安定」「反復」「停滞」の 3 状態に基づく

分類を行ったが，すべての学習行動がこれらに明確に分類できるわけではない．例えば，課題の序盤に見られる大量のコードを一気に記述するような行動（例：数十行の実装フェーズ）は，現在のいずれの定義にも当てはまらず，未分類となる場合がある．このような行動は，むしろ「生成」や「実装」といった新たな状態カテゴリとして捉えるべき可能性があり，状態定義の拡張とその分類ルールの設計が今後の検討課題である．また，現在のルールでは，特定の特徴量（例：lines_Changed_sum）がしきい値外となった場合にスコア加算が行われず，他の特徴量だけで分類が決定されるため，こうした未分類や誤分類のリスクも今後の精緻化において考慮すべきである．

6.3 おわりに

本研究は，プログラミング教育における学習過程の可視化と理解支援に向けた第一歩として，軽量かつリアルタイムに動作可能な学習状態分類手法を提案し，その有効性を初期的に示した．将来的には，本手法を発展させ，実データ分析や動的分類への対応，教育支援ツールとの統合を通じて，「個別最適な学び」の実現に資する知的学習支援の構築を目指す．

謝辞 本研究は，テレコム先端技術研究支援センター SCAT 研究の助成を受けたものです．

参考文献

[1] Amjad Altadmri and Neil C. C. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*, pp. 522–527, March 2015.

[2] Anthony Estey, Hieke Keuning, and Yvonne Coady. Automatically classifying students in need of support by detecting changes in programming behaviour. *Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE '17)*, pp. 189–194, March 2017.

[3] Ge Gao, Samiha Marwan, and Thomas W. Price. Early performance prediction using interpretable patterns in programming process data. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*, pp. March 13–20, March 2021.

[4] Paulo Blikstein. Using learning analytics to assess students' behavior in open-ended programming tasks. *Proceedings of the 1st International Conference on Learning Analytics and Knowledge (LAK '11)*, pp. 110–116, February 2011.

[5] Charis Charitsis, Chris Piech, and John C. Mitchell. Feedback on program development process for cs1 students. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE '22)*, p. 1150, March 2022.

[6] 藤原賢二, 上村恭平, 井垣宏, 吉田則裕, 伏田享平, 玉田春昭, 楠本真二, 飯田元. スナップショットを用いたプログラミング演習における行き詰まり箇所の特定. コンピュータソフトウェア, Vol. 35, No. 1, pp. 13–113, 2018.

[7] Tobias Kohn. The error behind the message: Finding the cause of error messages in python. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, pp. 524–530. ACM, 2019.

[8] 槇原絵里奈, 井垣宏, 吉田則裕, 藤原賢二, 飯田元. プログラミング演習における探索的プログラミング行動の自動検出手法の提案. コンピュータソフトウェア, Vol. 35, No. 1, pp. 1110–1116, 2018.