RK-001

絶対評価を用いたプログラミング演習における理解度測定
Evaluation of Understanding with Absolute Evaluation in Programming Exercise

小山 昂紘 † 谷川 紘平 † 原田 史子 ‡ 島川 博光 ‡
Takahiro Koyama Kouhei Tanigawa Fumiko Harada Hiromitsu Shimakawa

1. はじめに

大学の情報教育は，学生にコンピュータに関する基礎知識や概念を理解させ，活用できる能力を身に付けさせることを目標としている．プログラミング教育の初期段階では，学生にプログラムを作成する能力を習得させるために，プログラミングに関する講義と実際にソースコードを記述するプログラミング演習を設けている場合が多い．プログラミング演習では，学生は個人ごとで演習課題を解くためにソースコードの作成に取り組む．

演習課題で作成するプログラムは変数の型や条件分岐による処理の流れなど，複数の分野の知識を結合して構成される．プログラミング初心者である学生は複数の分野をしっかり理解せずに演習を進めていく場合がある．そのため，課題内容が複雑になると，理解が足りないことが原因で課題を解けないといった問題が発生する．

現在のプログラミング演習では，学生の作成したプログラムが与えられた課題に対して正しいデータを出力するかで課題内容を理解できたかを評価する．しかし，プログラムの実行結果からわかるのは学生のプログラミングに対する総合的な理解度であり，個々の分野に関する理解度を測ることはできない．本論文では，学生が作成したソースコードから分野ごとの理解度を絶対的に測る手法を提案する．本手法で求めた理解度を使うことにより学生の理解度に応じた学習指導が可能となる．

2. プログラミングに関する学生の理解度

2.1 理解度に応じた学習指導の必要性

演習課題で作成するプログラムは，複数の分野の知識を用いて構成される．プログラミング初心者である学生は自身がどの分野の理解が足りないのか判断できない．そのため，学生は自分で復習するべき内容を見つけることができない．また，学生は課題を解くことができないとき，指導者に対して質問をする．しかし，指導者も質問してきた学生がどの分野を理解できていないか判断できないため，適切な指導をすることは困難である．

課題を解くには，難易度に関わらず，さまざまなプログラミングに関する知識が必要とされる．各分野に関する理解度を分野別理解度と呼ぶ．学生の分野別理解度を測ることができれば，学生個人の理解度に応じた指導や自習項目の提示といった，きめ細かな学習指導が可能となる．

2.2 評価項目の細分化

学生が理解できていない分野を見つけるには，ソースコードを指導者が検査し，学生が理解できているか確認する必要がある．そのため，プログラミング演習の評価では，ソースコードを構成するための知識を学生が理解できたか検証するために，評価するべき項目を細かく設定する必要がある．個々の学生が理解できていない分野を明らかにするためには，複数の課題で各分野に関連する評価項目を設定し，それらを達成できているかを調べる必要がある．

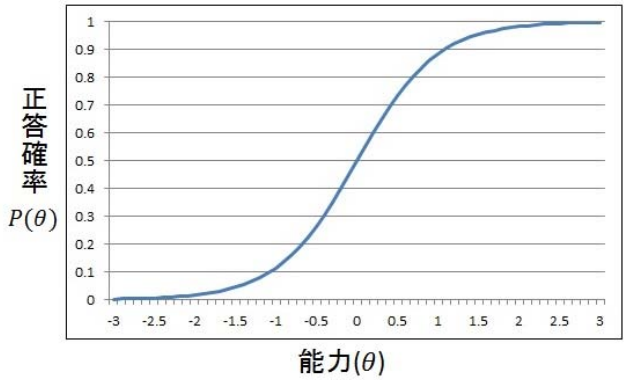

図 1: ロジスティック曲線

2.3 絶対評価の必要性

理解度に応じた学習指導は，多くの指導履歴を調べることで精度を高くできる．そのためには，他クラスの指導履歴や過去のクラスの指導履歴を利用することが考えられる．多数の学生が学ぶプログラミング演習においては，複数の指導者が指導にあたる．そのため，評価する指導者によって学生の理解度の評価値が変化してしまう．指導履歴を利用しても評価した理解度の基準が異なるため学生の理解度にあった適切な指導はできない．指導履歴を用いて学生の理解度に応じた学習指導をおこなうには指導者間で理解度を評価する基準を統一する必要がある．

指導者は学生に必ず解答してほしい課題に多く配点することが多い．そのため，簡単な課題が難しい課題よりも高く配点されている場合がある．よって，学生の理解度を指導者の配点に基づく点数で評価することはできない．また，偏差値を用いて理解度を評価すると母集団の能力により評価が変わってしまう．学生の理解度を評価するためには問題の配点や母集団に影響されず，絶対評価をすることが重要である．

2.4 項目反応理論

項目反応理論 [1] は評価項目を正解，不正解の採点結果から受験者の能力を絶対評価で測るテスト理論である．項目反応理論では，問題に正解する確率から受験者の能力を測る．そのため，項目反応理論で測った能力は問題の配点や母集団によって変わることがない．

項目反応理論では図 1 の曲線を問題ごとに求める．この曲線をロジスティック曲線という．ロジスティック曲

†立命館大学大学院理工学研究科
‡立命館大学情報理工学部

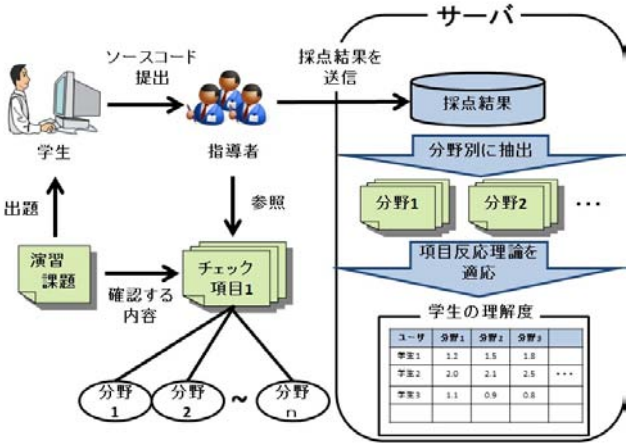

図 2: 手法の流れ

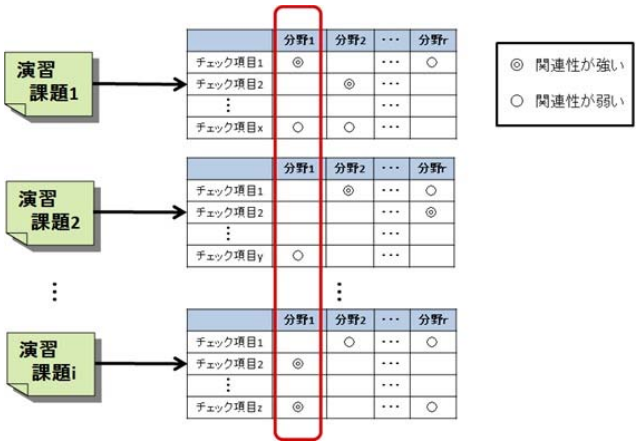

図 3: 分野 1 の理解度判定に用いられるチェック項目群

線は受験者の能力の違いにより問題に正答する確率を示している．この曲線は以下の 2 つのパラメータと式 1 を用いて求められる．

項目識別力 (a):能力値による差の付きやすさ
項目難易度 (b):問題の難しさ

$$P(\theta) = \frac{1}{1 + e^{-Da(\theta - b)}}, (-3 \leq \theta \leq 3) \qquad (1)$$

3. プログラミングで測る分野別理解度

3.1 プログラミング演習における学生の理解度測定

本論文では，プログラミング演習における学生の分野別理解度をもとめる手法を提案する．本手法では，学生が演習課題の内容を理解できたか確認するために，ひとつの演習課題に対して複数のチェック項目を設ける．次に，各チェック項目をプログラミングの理解度を表現する分野に関連付ける．複数のチェック項目に対する採点結果を分野別で抽出し，項目反応理論を用いて学生の分野別理解度を求める．

本手法の流れを図 2 に示す．本手法では，演習課題ごとに確認するべき内容をチェック項目として複数作成する．チェック項目はプログラミングに関する複数の分野に関連している．学生は課題を解くためにソースコードを作成し，指導者はチェック項目に従い，学生が作成したソースコードを採点する．多数の学生から提出されたソースコードを採点するために，複数の指導者がランダムに採点にあたるものとする．サーバ上ですべての課題のチェック項目に対する採点結果を分野別に抽出する．抽出したチェック項目に対する採点結果を分野別で項目反応理論に適応し，学生の分野別理解度を測る．

3.2 演習課題とチェック項目

演習課題とチェック項目の関係を図 3 に示す．はじめに，指導者は学生のプログラミングに対する理解度を表現する分野を複数想定する．次に，演習課題に対して複数のチェック項目を作成する．各チェック項目をそれぞれ，ひとつ以上の分野に関連付ける．チェック項目に関連する分野には関連性が強いものと弱いものがある．指導者は学生が作成したソースコードをチェック項目に従って採点する．ソースコードをチェック項目に従って採点をすることにより，指導者間で確認する内容を統一できる．一方，学生は複数の課題において，同一分野のチェック項目を正解することでその分野を理解したといえる．しかし，複数の指導者がいる場合，演習課題ごとに採点する指導者が変わる可能性が高い．そのため，特定の学生に対して，各分野をどの程度，理解しているか指導者が判断するのは困難である．そこで，本手法では、図 3 に示すように，分野が共通するものを複数の演習課題から抽出する．そして，抽出したチェック項目の正解，不正解を検査する．こうすることにより，複数の指導者が分担して演習課題を採点しても，対象となる学生の各分野に対する理解度を判定できる．

3.3 分野別理解度の測定

学生の分野別理解度をチェック項目に対する採点結果とロジスティック曲線の値から測る．学生の理解度 θ を求める計算式を式 2 に示す．ϵ は学生のチェック項目に対する採点結果であり，チェック項目に正解したなら 1，間違えたなら 0 とする．n はチェック項目の数で，$P_k(\theta)$ は理解度 θ の学生が k 番目のチェック項目に正解する確率を示す．

$$\begin{aligned} \epsilon &= \{\epsilon_1, \epsilon_2, \cdots, \epsilon_n\} \\ \epsilon &\in \{0, 1\} \\ A_\theta &= \prod_{k=1}^{n} \{P_k(\theta)\}^{\epsilon_k} \{1 - P_k(\theta)\}^{1-\epsilon_k} \end{aligned} \qquad (2)$$

実際に表 1 を例に，学生の理解度を測る例を示す．表 1 に各チェック項目のロジスティック曲線の値と採点結果が $\epsilon = \{1, 1, 0\}$ となった時の計算結果を示す．最も高い確率を示した θ の値が学生の理解度となる．表 1 の内容では $\theta = 1$ の時が最も高い確率を示したので学生のこの分野に対する理解度は 1 となる．

4. プログラミング演習における実験

4.1 演習形態

本手法を用いてプログラミング演習における学生の分野別理解度を測ることができたか検証した．演習は全部

表 1: チェック項目の正答率と計算結果

θ	項目 1	項目 2	項目 3	(1，1，0)
-3	0.15	0.04	0.01	0.00594
-2	0.3	0.12	0.03	0.03492
-1	0.5	0.27	0.08	0.1242
0	0.7	0.5	0.23	0.2695
1	0.85	0.73	0.5	0.31025
2	0.93	0.88	0.77	0.188232
3	0.97	0.96	0.92	0.074496

表 2: 分野名と内容例

分野名	内容例
コンピュータの概念	ディレクトリの概念
プログラミングの環境	コンパイルの仕方
プログラムの流れ	条件分岐・繰り返しの流れ
値の評価	比較演算子，算術式
メモリについて	配列の確保
変数の型	double 型，int 型
プログラミングの文法	条件分岐の書き方
ライブラリ関数	出力の関数
ユーザビリティ	入力を促す表示
開発過程	デバッグ処理
デザイン	コメント，インデント

評価基準	チェック項目	コンピュータの概念	プログラムの流れ	変数の型	プログラムの文法	デザイン
float型，もしくはdouble型の変数を定義している	体重，身長，標準体重を計算するための浮動小数点型の変数を定義している			◎	○	○
標準入力している	身長と体重を 標準入力 入力し変数に代入している	○			◎	
if文で判断している	体重と標準体重の差が8.0kgより大きいかif文で判断しているか		◎		○	
else-if文で判断している	体重と標準体重の差が-8.0kgより小さいか else-if文で判断しているか		◎		○	
else文で処理している	上記以外の場合の処理をelse文で処理しているか		◎		○	
字下げとコメントを付ける	インデントや処理を示すコメントがある					◎

図 4: チェック項目例

図 5: サーバ 1 の相関

で 15 週あり，はじめの 2 週間はエディタの使い方やコマンドプロントの使い方など，演習をするために必要な基礎知識を教えた．また，9 週目と 15 週目はプログラミングのテストを実施した．演習課題を難易度別に基本・応用・発展の 3 段階にわけた．学生には各週で演習課題を与え，プログラムを作成させた．

学生は専用の Web サイトを使用して演習に取り組む．学生は作成したソースコードをオンライン上に提出する．指導者はチェック項目に従ってソースコードを採点する．採点結果はデータベース上に保存される．今回は表 4.1 に示す 11 分野を想定してチェック項目を作成した．

演習課題の例を以下に示す．

> 入力された身長と体重から指定された式を用いて標準体重を計算し表示するプログラムを作成する．ただし，体重が標準体重よりも 8kg 軽ければ「少し痩せています」，8kg 重ければ「少し太っています」と警告する機能を設ける．

この課題に対して図 4 のチェック項目を作成する．チェック項目に対して関連がある分野にはマークを付ける．

4.2 ロジスティック曲線の作成

パラメータの推定は項目反応理論の中で最も計算量の多い部分である．そのため，専用のソフトウェアを利用して求めるのが一般的である．今回は EasyEstimation[2] をパラメータ推定ソフトウェアとして使用した．パラメータ推定のために使用する学生の採点結果は課題が出題されてから 1 週間以内の採点結果を使用した．これは，課題の提出に期限がないため，演習が進んでから過去の課題を解く学生がいると正確なパラメータを推定できないためである．学生の理解度 θ の値は-3〜3 の間で 0.1 刻みとした．

4.3 分野別理解度の計算

基本・応用の演習課題で各学生の分野別理解度を測った．また，チェック項目に関連する分野として，関連性の強い分野に着目した．理解度を求める週までに出題された課題を評価対象とした．出題された課題のチェック項目を分野別に分ける．分野別に分けたチェック項目の採点結果から 3.3 節で示した計算方法で学生の分野別理解度を求める．

5. 評価

5.1 実験結果

本手法により理解度を正しく測れるか検証するために，コンピュータの概念，プログラミングの文法，デザイン，開発過程の 4 分野について評価実験をした．チェック項目数は演習課題全体でコンピュータの概念が 51 項目，プログラミングの文法が 104 項目，デザインが 62 項目，開発過程が 17 項目あった．理解度を測る学生がソースコードを提出しなかった課題に対しては，理解が足りないためプログラムを作成することができないと仮定し，課題に対するすべてのチェック項目を間違え扱いとした．学生の分野別理解度を 2 台のサーバの採点結果から評価をした．

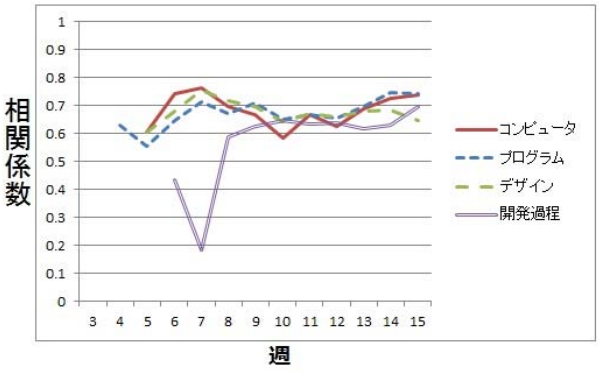

図 6: サーバ 2 の相関

学生の分野別理解度と発展課題のチェック項目に対する正答率の相関を調べた．図 5・6 に各週の相関をグラフとして示す．1 つのサーバにつき約 95 人の学生のデータを使用している．発展問題を解く学生がいないことや，対応する分野のチェック項目が無いことから演習の開始時は相関を調べることができなかった．コンピュータの概念，プログラミングの文法，デザインの 3 分野に関しては各週で安定して強い相関を示した．しかし，開発過程に関しては 8 週目以前の相関はあまり見られなかった．

5.2 考察

学生の分野別理解度と発展課題のチェック項目に対する正答率は，コンピュータの概念，プログラミングの文法，デザインの 3 分野で常に高い相関を示した．開発過程に関しては，評価対象のチェック項目数が増加すると相関が現れた．このことから，学生の理解度を測るためには，一定の問題数が必要であることがわかる．また，プログラミングの文法は相関が出た他の 2 つと比べてチェック項目の数が多かったが，相関に明確な違いは見られなかった．よって，チェック項目数が定数を超えれば相関が現れ，チェック項目数がそれ以上増加しても理解度を測定する能力に変化はないと考えられる．本実験の環境では，チェック項目が全体で 50 項目程度あれば学生の理解度を測定するのに十分であるといえる．

学生の課題の提出状況を調べると 4 週目と 13~15 週目が他の週に比べて課題の提出数が多かった．4 週目に関しては，3 週目に提出数が少ないため，3 週目の課題を 4 週目に提出したと考えられる．これは，学生が演習の環境に慣れていなかったためと考えられる．13~15 週目に関しては演習の終了時期が近いため，今まで提出していなかった課題を提出したためと思われる．このことから，13~15 週目の分野別理解度は強い相関が出ていたが、信頼性が薄いと考えられる．

以上のことから，4 週目，13~15 週目を除いては各週で学生の分野別理解度を測ることができたと考えられる．本実験により学生の分野別理解度をより正確に測るためには，提出期限を明確に設定する必要があることがわかった．また，チェック項目の数が一定数を越えると理解度を測る能力に変化がないことがわかった．

各分野のチェック項目は毎週ある程度の数だけ用意すれば，演習が進むことで変化していく学生の分野別理解度を測れる．また，特定の期間，例えば 5 週程度を設定し，期間内で提出された課題から理解度を計算した方が，過去のチェック項目に影響されずに，その時の学生の理解度を評価できる．

5.3 既存研究

学生のプログラミングに関する分野ごとの理解度を測る既存研究として文献 [3] が挙げられる．文献 [3] では，学生のプログラミングに関する理解度を学生へのアンケートから求めている．この研究では，プログラム制御文について理解できたかアンケートをしている．しかし，アンケートによって測る理解度は学生自身の自己評価によるものである．そのため，学生が実際には理解していなくても理解したと思っている場合が考えられる．

プログラミングの課題に対してチェック項目を作成し，学生の理解度を測る既存研究として文献 [4] が挙げられる．文献 [4] では，指導者がチェック項目に従って学生に質問し，学生の回答結果を評価基準に従って 5 段階で評価をする．しかし，この研究では学生のプログラミングに関する総合能力は測ることができるが，学生の分野別理解度は測ることができない．

6. おわりに

本論文では，プログラミング演習の課題に対してチェック項目を設け、項目反応理論を用いて学生の分野別理解度を絶対評価で測る手法を提案した．評価対象とする演習課題に対するチェック項目を分野別に抽出し，各チェック項目に対する採点結果から学生の分野別理解度を測った．学生の分野別理解度を利用することにより，多くの指導履歴を用いて学生にきめ細かな学習指導ができる．

提案手法の有用性を検証するために，本手法を適応したプログラミング演習を実施し，学生の分野別理解度を求めた．求めた学生の理解度と課題に対する採点結果を調べるとコンピュータの概念，プログラミングの文法，デザインについて強い相関がみられた．これにより，本手法を用いることにより，プログラミング演習における学生の分野別理解度を測ることができたと考える．

今後の課題として評価期間の設定，関連性の弱い分野についての検証が挙げられる．

参考文献

[1] 高橋 正見，項目反応理論入門　-新しい絶対評価-，イデア出版，2002

[2] 熊谷 龍一，初学者向けの項目反応理論分析プログラム EasyEstimation シリーズの開発，日本テスト学会誌，5，pp.107-118，2009

[3] 佐野 繭美，橋本 はるみ，河俣 英美，横山 宏，松永 公廣，授業支援システムを用いたプログラミング基礎教育の授業実践， 摂南大学教育学研究 4，pp.43-56，2008

[4] 松浦 佐江子，プログラミング演習における評価方法の改善，論文誌 IT 活用教育方法研究，Vol.8，No.1,pp.51-55，(社) 私立大学情報教育協会，2005