The effect of deductive and inductive teaching methods in an introductory programming course

Dr. Jen Symons, University of Portland

Jen Symons is an Assistant Professor in Biomedical and Mechanical Engineering in the Shiley School of Engineering at the University of Portland. She is most passionate about teaching biomechanics and statistics for engineers. Her research focuses on understanding the causes of musculoskeletal injury and developing noninvasive mechanisms that prevent injuries and/or enhance performance in equine athletes.

Dr. Heather Dillon, University of Portland

Dr. Heather Dillon is an Associate Professor in Mechanical Engineering at the University of Portland. Her teaching focuses on thermodynamics, heat transfer, renewable energy, and optimization of energy systems. She currently leads a research team working on energy efficiency, renewable energy, and fundamental heat transfer. Before joining the university, Heather Dillon worked for the Pacific Northwest National Laboratory (PNNL) as a senior research engineer.

Dr. Joseph P Hoffbeck, University of Portland

Joseph P. Hoffbeck is a Professor of Electrical Engineering at the University of Portland in Portland, Oregon. He has a Ph.D. from Purdue University, West Lafayette, Indiana. He previously worked with digital cell phone systems at Lucent Technologies (formerly AT&T Bell Labs) in Whippany, New Jersey. His technical interests include communication systems, digital signal processing, and remote sensing.

The Effect of Deductive and Inductive Teaching Methods in an Introductory Programming Course

Abstract

Many first- and second-year engineering students are required to complete an introductory programming course as part of their program. Typically, class time within these courses is divided into traditional instructor lecture and student work time interacting with the software. This study aimed to determine the effect of different instructor teaching methods on student quiz performance as it related to challenging concepts (i.e. functions, conditional statements, and loops). A Latin square design was used to assess 3 different teaching methods across 3 course sections of students, all taught by the same instructor. Teaching methods included a single deductive lecture explaining the concept, a similar deductive approach that divided content into two smaller lectures within one session, and an inductive approach that presented code within an applied context. After the teaching methods were delivered, points from quizzes completed by students were divided into categories based on the three course concepts. The effect of teaching method on percentage of points earned by students for each concept was assessed by ANOVA. Teaching method showed a trend, but failed to achieve statistical significance (p=0.11). However, post-hoc tests indicated a significant difference between the control teaching method and the inductive teaching method (p=0.04). The control teaching method yielded the best student quiz scores, whereas the inductive teaching method yielded the worst scores. These results support a deductive approach when teaching first and second year students introductory programming skills, as opposed to an inductive approach that may be more well suited outside foundational engineering courses.

Introduction

Engineering educators are required to teach potentially complex concepts to a diverse group of students with varying skills and knowledge. Prior researchers have studied the learning behavior and patterns unique to computer programming [1-4]. Within the context of an introductory programming course, choosing the most effective teaching method can be challenging. The analogy of learning a new language is sometimes used to describe the types of skills that are needed to communicate with a computer. Language pedagogy may prove valuable in introductory programming courses. Current practices in language instruction advise deductive teaching for simple rules and less skilled learners, versus inductive teaching for complex rules and more skilled learners [5]. Deductive teaching begins with a general rule of theory presented by the instructor, followed by specific examples and practice problems. Inductive teaching begins with specific examples presented by the instructor and requires students to make observations to formulate a general rule or theory.

The use of inductive teaching methods has been studied for many engineering and computer science applications [6]. Inductive methods are widely accepted as beneficial to students in engineering and computer science, particularly for promoting deeper understanding and developing critical thinking skills. However, the implementation and background knowledge of students may play a role in the success of these methods. Inductive teaching requires students to make observations and connections to programming concepts within the context of a problem. Student ability to make observations and connections is dependent on prior knowledge. Introductory programming courses rarely have prerequisites. Therefore, not all students have basic vocabulary and understanding of programming concepts[7], which makes inductive learning a unique challenge within these types of courses.

Many prior research groups have studied pedagogical methods for introductory programming courses. Medeiros provided a summary of prior studies on this topic [7]. Hoffbeck et al. reported on the development of an introductory programming course using MATLAB [1]. Selby mapped the types of programming concepts typically used in a course of this type to Bloom's taxonomy [8]. A summary of recent work using different teaching methods for programming courses is shown in Table 1. Jonsson reported significant improvement in end of course examinations for students using a flipped classroom approach [9]. Lykke et al. studied the emotional response of students to different types of education methods in an introductory programming course [2]. The only introductory programming course that focused on inductive learning methods did not formally assess students in the study [4]. Instead, the authors focused on outlining the specific techniques and methods used.

Table 1: Summary of interdisciplinary laboratory modules from the literature.

Author	Year	Course Type	Teaching Methods
Lykke [2]	2015	Undergraduate Intro Programming	Lecture and Problem Based Learning
Bogdanovych and Trescak[3]	2015	Undergraduate Intro Programming	Gamification
Jonsson [9]	2015	Undergraduate Intro Programming	Flipped classroom, peer discussion, and just-in-time teaching
Tom [10]	2015	Undergraduate Intro Programming	The Five C Framework
Sedelmaier and Landes [4]	2015	Undergraduate Intro Programming	Inductive learning
Hoffbeck et al. [1]	2016	Undergraduate Intro Programming	Problem Based Learning
Spangsberg and Brynskov [11]	2017	Graduate Intro Programming	Code labeling
This Study	2019	Undergraduate Intro Programming	Lecture and Inductive methods

No prior studies focused on comparing traditional lecture (i.e. deductive) and inductive methods for undergraduate student in an introductory programming course, while also controlling for the same instructor, a potentially large confounding factor. This study explored the use of deductive and inductive teaching applied to challenging concepts like functions, conditional statements, and loops. These concepts were presented during the middle of the term (weeks 7-9). Based on the fact that all students were given 6 weeks of instruction explaining basic programming vocabulary and concepts, we hypothesized that an inductive teaching approach would benefit student performance when assessing more challenging concepts in an introductory programming course. In this study, we have taken the knowledge of prior groups and tested the concept of when to implement inductive learning in a well-controlled experiment.

Methods

All study protocols were approved by the University of [Blinded] Institutional Review Board. First-year engineering students (n=85) in 3 sections of an introductory MATLAB programming course for non-computer science majors were invited to participate based on a common instructor of record. A subset of these students (n=66) provided informed consent to allow their coursework to be included in this study anonymously. Students of varying skill level and prior knowledge were considered to be randomly distributed across course sections.

A Latin square design [12] was used to assess the effect of 3 different teaching methods on student performance. A Latin square design typically incorporates 3 factors: one main factor and two nuisance factors. In this case, teaching method was the main factor of interest because it could be modified by the instructor. Teaching methods included a single 10-15 minute deductive lecture (control), two 5-7 minutes deductive lectures separated by computer work time (iteration), and an application-based approach that described a problem and provided a solution code (inductive). Course section and course topic were considered nuisance factors because their levels could not be changed or controlled by the instructor. Affected course topics included functions, conditional statements, and loops.

Three class sessions were modified in each course section (i.e. 9 total sessions across 3 sections). Each of the 9 modified class sessions included a unique combination of teaching method and course topic, while ensuring all 3 topics and all 3 teaching methods were delivered in each course section (Table 2). For example, the instructor used the control teaching method to explain functions for students in the first course section. Then, the instructor used the inductive teaching method to explain functions for students in the second course section and the iteration teaching method to explain functions for students in the third course section. This approach allowed for all teaching methods to be evaluated in combination with all course topics, as well as all course sections. This mitigated any potential confounding effect of preferentially pairing certain teaching methods with certain course topics and/or certain course sections.

Table 2: Latin square design indicating teaching method used for each combination of course section and course topic.

	Topic		
Section	Functions	Conditional Statements	Loops
1	Control	Iteration	Inductive
2	Inductive	Control	Iteration
3	Iteration	Inductive	Control

Following teaching method deliveries, students in all sections completed a series of 2 quizzes (20 points each). Each quiz included problems assessing multiple course topics. Points for all quiz problems were summed in 3 categories based on course topics assessed (i.e. functions, conditional statements, and loops). Total points for each category were of similar magnitude, but not equal. Therefore, student points earned in each category required normalization prior to statistical comparison (i.e. divide points each student earned within the category by total points possible within the category). Thus, points earned by each student within each category (i.e. functions, conditional statements, and loops) were normalized to the total points possible within each category. Normalized student scores represented the percentage of points earned by each student within each course topic. The effect of teaching method on normalized student score was evaluated by a one-way analysis of variance (ANOVA), with teaching method as the single factor (SAS 9.4, PROC MIXED).

Results

Within the one-way ANOVA, the effect of teaching method on student performance showed a statistical trend, but failed to indicate statistical significance (p=0.11). However, post-hoc comparisons yielded a statistically significant difference (7%, p=0.04) between 2 of the teaching methods, the control deductive method and the inductive approach (Figure 1).

The control teaching method, the single 10-15 minute deductive lecture, produced the greatest student scores with a least square mean of 73% and a standard error of 3%. The inductive method produced the lowest student scores with a least square mean of 66% and a standard error of 3%. The iterative method of 2 shortened lectures separated by computer work time produced intermediate student scores with a least square mean of 71% and a standard error of 3%.

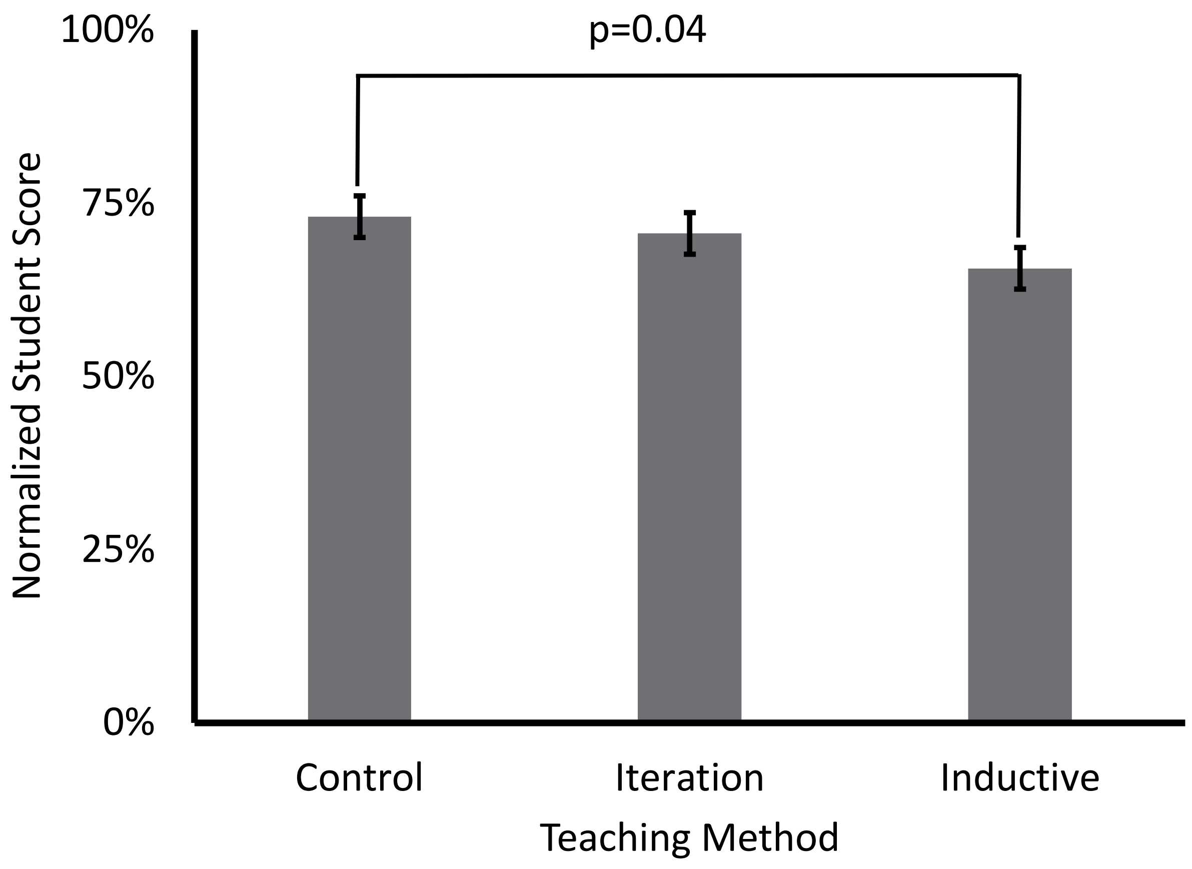

Figure 1: Histogram comparing student performance (i.e. normalized student score) aggregated for all course topics as a result of different teaching methods.

Discussion

This study aimed to understand the effect of inductive teaching, compared to deductive teaching, in an introductory programming course. Previous studies in engineering and language pedagogy support the use of inductive teaching [1-2]. We hypothesized that inductive teaching would improve student performance related to more complex topics like functions, conditional statements, and loops. However, the results of this study suggest a different conclusion. Student performance was highest when concepts were taught using deductive methods, specifically a single lecture. It appears that inductive teaching may be less effective for students learning these challenging concepts in an introductory programming course.

Many of the students in an introductory programming class may lack the basic vocabulary and concepts for programming needed to make observations, formulate theories, and thrive with inductive teaching approaches. This difference in prior student knowledge increases the cognitive demand for students to simultaneously construct foundational vocabulary and concepts, while also recognizing these components within problems or applications. For this reason, inductive teaching may be less effective, and potentially negative, for first-year students with little or no background knowledge in programming. In this case, students are attempting to construct new conceptual knowledge on limited or absent foundational knowledge, similar to building on quicksand.

Poor student performance associated with inductive approaches indicates that introductory students may require more scaffolding in their instruction. This study supports instructors using a traditional, deductive approach to help students build a solid foundation (basic vocabulary and concepts) that may be more effective toward improving student performance and learning. Because traditional, deductive teaching methods are instructor-centered, they allow the teacher to assist students in linking key concepts to programming constructs in a hierarchal way.

Limitations of the present study are centered around uncontrollable nuisance factors that may have influenced student performance, namely student distribution between sections and course topics evaluated. Within every student population, there is a range of abilities influenced by prior experiences and education. This student population of varying abilities was considered to be randomly distributed between the 3 course sections. However, one course section may have had a disproportionately skilled or unskilled group of students. This disproportionate distribution would influence the course section least square mean of student score. However, it would not likely influence the least square mean of student score based on teaching method. The least square mean of student score for each teaching method is aggregated from student scores from all 3 sections, as well as all 3 course topics. For instance, the least square mean of student score for the control method incorporates scores from the first section on functions, the second section on conditional statements, and the third section on loops. Therefore, student scores from a disproportionately skilled or unskilled group of students will comparably influence least square means for each of the teaching methods. Similarly, the course topics evaluated may have resulted in different student scores, depending on ease or difficulty of concepts. However, course topics would not likely influence the least square mean of student score based on teaching method. The least square mean of student score for each teaching method is aggregated from student scores from all 3 course topics, as well as all 3 sections. The Latin square design used in this study is useful in mitigating, or at least applying equally, the effects of uncontrollable nuisance factors like student population and course topics required by the course.

In the same way that deductive and inductive teaching may be suited for simple and complex concepts, respectively, different teaching approaches may be needed for different populations of students [5]. The observed cohort was composed of engineering students outside of a computer science program. This cohort was hypothesized to have minimal prior programming knowledge, which was not assessed. However, this information is likely influential in determining the most effective teaching methods. Future studies may be designed to understand how teaching methods may be tailored to different populations of students. For example, in a larger cohort, students could complete a pre-test to determine prior knowledge. A similar method of deductive and inductive approaches could be applied to students with and without a large existing vocabulary and advanced knowledge. Future studies may also extend these research methods and their applications to other foundational engineering courses.

Conclusions

Students in an introductory programming course performed best when taught challenging concepts using a deductive approach, compared to an inductive approach. Even though students were assessed after 6 weeks of basic instruction in programming, they still benefited from a traditional lecture, rather than an application-based approach. Therefore, deductive teaching should continue through the duration of introductory programming courses. Inductive approaches may be better suited for more advanced engineering courses that allow students to draw from a deeper collection of skills and knowledge.

References

[1] J. P. Hoffbeck, H. E. Dillon, R. J. Albright, W. Lu, and T. A. Doughty, “Teaching programming in the context of solving engineering problems,” in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016, pp. 1–7.

[2] M. M. C. S. N. Lykke, “Motivating Students through Positive Learning Experiences: A Comparison of Three Learning Designs for Computer Programming Courses.,” *J. Probl. Based Learn. High. Educ.*, vol. 3, no. 2, pp. 80–108, 2015.

[3] A. Bogdanovych and T. Trescak, *Teaching Programming Fundamentals to Modern University Students*. 2016.

[4] Y. Sedelmaier and D. Landes, “Active and Inductive Learning in Software Engineering Education,” in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, 2015, pp. 418–427.

[5] R. Ellis, “Current Issues in the Teaching of Grammar: An SLA Perspective,” *TESOL Q.*, vol. 40, no. 1, pp. 83–107, 2006.

[6] M. J. Prince and R. M. Felder, “Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases,” *J. Eng. Educ.*, vol. 95, no. 2, pp. 123–138, 2006.

[7] R. P. Medeiros, G. L. Ramalho, and T. P. Falcao, “A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education,” *IEEE Trans. Educ.*, pp. 1–14, 2018.

[8] C. C. Selby and C. C., “Relationships: computational thinking, pedagogy of programming, and Bloom’s Taxonomy,” in *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE ’15*, 2015, pp. 80–87.

[9] H. Jonsson, “Using flipped classroom, peer discussion, and just-in-time teaching to increase learning in a programming course,” in *2015 IEEE Frontiers in Education Conference (FIE)*, 2015, pp. 1–9.

[10] M. Tom, *Five Cs Framework: A Student-centered Approach for teaching programming courses to students with diverse disciplinary background*, vol. 8. 2015.

[11] T. Spangsberg and M. Brynskov, *Code-labelling: A teaching activity encouraging deep learning in a non-STEM introductory programming course*. 2017.

[12] W. D. Wallis and J. C. George, *Introduction to Combinatorics*, 2nd Editio. CRC Press, 2017.