Can You Teach Me To Machine Learn?

An Exploration of Pedagogical Content Knowledge for Teaching Machine Learning to Non-Majors

Elisabeth Sulmont
Centre for Intelligent Machines
McGill University
Montreal, Canada
elisabeth.sulmont@mail.mcgill.ca

Elizabeth Patitsas
Integrated Studies in Education
McGill University
Montreal, Canada
elizabeth.patitsas@mcgill.ca

Jeremy R. Cooperstock
Centre for Intelligent Machines
McGill University
Montreal, Canada
jer@cim.mcgill.ca

ABSTRACT

Machine learning (ML) has become an important topic for students across disciplines to understand because of its useful applications and its societal impacts. At the same time, there is little existing work on ML education, particularly about teaching ML to non-majors. This paper presents an exploration of the pedagogical content knowledge (PCK) for teaching ML to non-majors. Through ten interviews with instructors of ML courses for non-majors, we inquired about student preconceptions as well as what students find easy or difficult about learning ML. We identified PCK in the form of three preconceptions and five barriers faced by students, and six pedagogical tactics adopted by instructors. The preconceptions were found to concern themselves more with ML's reputation rather than its inner workings. Student barriers included underestimating human decision in ML and conflating human thinking with computer processing. Pedagogical tactics for teaching ML included strategically choosing datasets, walking through problems by hand, and customizing to the domain(s) of students. As we consider the lessons from these findings, we hope that this will serve as a first step toward improving the teaching of ML to non-majors.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Computing methodologies** → *Machine learning*;

KEYWORDS

Machine learning; computer science education

ACM Reference Format:
Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R. Cooperstock. 2019. Can You Teach Me To Machine Learn?: An Exploration of Pedagogical Content Knowledge for Teaching Machine Learning to Non-Majors. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287392

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02…$15.00
https://doi.org/10.1145/3287324.3287392

1 INTRODUCTION

Not long ago, machine learning (ML) was a topic of interest limited largely to computer science (CS) and statistics. But with the steady increase of accessible computing power and large data sets, ML has been applied to a wide range of domains. With its increasing use, ML has become relevant to a much broader community, and with it a need to understand how we can teach ML to non-major students.

Furthermore, when we consider the implications of ML, such as, job automation [6, 10], privacy concerns [4, 28], and algorithmic bias perpetuating discrimination [9, 25], ML is a critical issue in today's society. Yet, the inner workings of ML remain a mystery to most people [26]. Given the ubiquity and increasingly important role of ML in consequential decision-making, we suggest that rectifying this lack of understanding is of high value to society.

Research in education has identified *pedagogical content knowledge* (PCK) as an important component of effective pedagogy [33]. PCK refers to the knowledge needed by an educator to teach a particular subject matter. Common types of PCK include an educator's knowledge about student misconceptions, preconceptions, barriers to learning, and tactics for assisting student learning.

1.1 Research Questions and Goals

In this paper, we explore examples of PCK for teaching ML to non-majors. For the purposes of this paper, we define "non-majors" as students without a background in CS or statistics.

Specifically, we wanted to know what preconceptions instructors can identify non-major students as having about ML, what misconceptions these students have, what barriers students face, and what pedagogical tactics are used to assist the students.

This is an exploratory work. The motivation for our project is to widen the literature on ML education for non-majors, specifically in how to teach the topic, which we believe is essential for ML education to develop.

Our contributions include identifying common preconceptions and barriers of students, as well as tactics of instructors. In our interviews with ten experienced ML instructors, our participants noted that students were generally unaffected by *mis*conceptions, but rather a lack of conception to begin with.

In previous work, using the same interview data, we identified and classified learning goals for non-majors using the Structure of Observed Learning Outcomes taxonomy (SOLO) [2].

Building on our previous work, in writing this paper we found it useful to organize the reported student barriers and instructor tactics around the SOLO taxonomy. What barriers did students face at each SOLO stage, and what tactics do educators use to target students at that stage?

Table 1: Interview Participants

ID	Position	Setting	Teaching
P1	Information Professor	Public Research University	Introductory information science
P2	Psychology Professor	Public Research University	Computational psychology
P3	Psychology Professor	Public Research University	Computational psychology and neuroscience
P4	Statistics Teaching Fellow	Public Research University	Intro ML for professional master's in data science
P5	Business Management Professor	European Private Catholic University	Introduction to business analytics for law students
P6	Biology Professor	Public Research University	Quantitative methods for biology
P7	Journalism Professor	Private Research University	Data and computational journalism
P8	Cognitive Science & CS Professor	Private Liberal Arts College	Topics in AI
P9	Data Scientist and Instructor	AI Research Center	Workshops for upper-level business people
P10	Data Scientist and Instructor	Online Learning Platform	ML courses for research scientists

2 RELATED WORKS

2.1 Pedagogical Content Knowledge

2.1.1 Origin. In 1986, Shulman argued that there was too strong of a separation between content and pedagogy in education research and policy, arguing that for a teacher to be effective, one was useless without the other. He proposed "pedagogical content knowledge" as a type of content knowledge specifically for teaching in an effort to emphasize and explain how a teacher transforms subject matter to be more amenable to students [33].

Shulman offered that PCK consisted of the following: "the most useful forms of representation of those ideas, the most powerful analogies, illustrations, examples, explanations, and demonstrations –in a word, the ways of representing and formulating the subject that make it comprehensible to others. . . also includes an understanding of what makes the learning of specific topics easy or difficult: the conceptions and preconceptions that students of different ages and backgrounds bring with them to the learning of those most frequently taught topics and lessons."[33, 6-7]. Although the concept of PCK was subsequently embraced by the research community, Shulman did not give a clear way to capture PCK with its broad definition. Difficulties in articulating and documenting PCK have been found since its introduction [1, 20, 23].

2.1.2 In Computer Science. Saeli et al. [31] conducted a review of PCK for programming. To define PCK-content, it was driven by four questions: why teach programming, what should be taught, what are the student difficulties, and how should the topic be taught. Hubbard's [17] review analyzed how computing PCK is conceptualized and investigated. Yadav et al. [37] measured PCK through teachers' responses to teaching vignettes about student's understanding of programming constructs, to which they found improvements needed in how teachers address students' programming errors. Other PCK research has focused on K-12 teacher education (e.g. [18, 24]). CS PCK education literature is limited, but there is work on student misconceptions for topics such as, programming (e.g. [7, 19, 30]), and algorithms and data structures (e.g. [8, 29, 32]).

2.1.3 In Statistics. Groth [12] sketched a hypothetical descriptive framework of statistical knowledge for teaching based off of research on mathematical knowledge for teaching [14–16], while maintaining the separation between statistics and mathematics. Groth cites building data and designing a study as examples of non-mathematical statistical knowledge. Lee and Hollebrands's [22] work on effectively engaging students in learning statistics with technology resulted in a framework for a specialized knowledge called technological pedagogical statistical knowledge (TPSK). There are three components in TPSK and each depend on the one before it: statistical knowledge, technological statistical knowledge, and technological pedagogical statistical knowledge.

2.2 Machine Learning Education Research

This area of research is still growing, as ML is such a new subject in university curricula. There has been work in AI education; notably the Symposium on Educational Advances in Artificial Intelligence, which first met in 2010. Research on ML education has been present through that venue, including assignment designs [13, 34].

Of relevance to our study is Lavesson's [21] case study of a master's level ML course (30 students). In the course evaluation results, Lavesson reports that students found the evaluation and comparison of learning algorithms (through statistical tests and different evaluation metrics) to be difficult. Students felt more comfortable comparing algorithms theoretically than comparing them empirically with statistical tests and experiments. The implementation of ML algorithms using programming was found to be easiest out of all learning outcomes. Our present work extends his findings by a more in-depth analysis of why certain topics are more difficult and with a focus on non-majors. Additionally, as opposed to sampling only a single course, our work contributes an analysis of data sampled over multiple courses within different disciplines.

2.2.1 Teaching Machine Learning to Non-Majors. Literature on teaching ML to non-majors is even more sparse than the already minimal literature on ML education, comprising of papers on curricula for non-majors. Way et al. [35] [36] presented their findings from a multi-year project called "Broader and Earlier Access to Machine Learning" to which resulted in an online repository of flexible module materials. Gil [11] designed a data science course with ML topics for non-programmers, through a workflow paradigm and a visual interface called WINGS. Pre-tested on non-majors, Gil proposes that learning basic ML concepts without programming is more approachable due to less time investment (e.g., required CS courses, lessons spent on programming).

3 METHODS

With no prior work focused on ML PCK, our goal for this study was to begin this line of investigation. We chose to speak to instructors who have taught non-majors to uncover the breadth of ML PCK.

3.1 Interviews

3.1.1 Participants. Participants were recruited through cold emails and mailing lists related to CS and data science education. The only requirement for participation was experience teaching ML concepts to non-majors. All of the participants' institutions reside in Canada or the USA, with the exception of P5 (Europe). All are professors, except for three instructors (P4, P9, P10) who all have their PhDs. Table 1 provides an overview of the participants.

3.1.2 Structured Interview. The development of the interview questions was informed by PCK literature. This affected questions intended both to probe for, and address, the difficulties and misconceptions faced by instructors and students. Appendix A contains the interview questions. All interviews were conducted over voice-call, except for P2 who answered by email. The average length of voice-call interviews was 21 minutes. Interviews took place between June and July 2018. Transcripts contained mostly paraphrased content of what the interviewees said, with some direct quotations. The first author conducted all the interviews and transcription.

3.2 Qualitative Analysis

Transcripts were coded iteratively for content, and were coded through consensus by four people. Two of the coders are authors of the paper; the other two are members of the same research group. All coders have a strong CS background, although one coder had no experience with ML. Two coders have strong education background and the other two have intermediate education backgrounds.

First, all four coders read the transcripts in their entirety and based on first impressions created four initial categories. These were preconceptions, easy for students, difficult for students, and tactics. Early in the coding process, we recognized the need for two more categories: hard to teach, and teacher goals. We free coded the transcripts in order of the questions (i.e., code all the transcripts for Question 2, then all the transcripts for Question 3, etc.).

3.2.1 Grouping Codes. Next, we clustered the codes that were repetitions of the same concept or very close to being repetitions. Through consensus, we then formed groups of codes that naturally fit together, regardless of the category under which they were originally coded, which resulted in 38 groups. Groups titled "Math", "Visual", "Human vs. Computer", and "Related to Domain" were the most populated groups. In addition, there were groups related to pedagogy ("Tools", "Techniques") and groups about preconceptions about ML. "Math" and "Programming" were unique groups because they encapsulated difficulties within their disciplines that did not exclusively occur while learning ML (e.g., debugging).

In examining our groupings, we noticed that the group titled "Design Decisions" was mostly full of "difficult for students" codes, whereas the group on teaching students a given ML algorithm was dominated by the "easy for students" codes. On further examination of the codes that represented some learning goal/objective, we observed a pattern that the easy to teach learning goals were aligned with the lower levels of the SOLO taxonomy and the hard to teach learning goals were grouped along with issues that correspond to higher levels of the SOLO taxonomy.

3.2.2 Regrouping Codes. Once we realized that the SOLO taxonomy could help us re-cluster our codes, we grouped our codes into nine categories: one for each SOLO stage as well as one for each transition between stages. Unlike the other SOLO stages, the pre-structural group did not have learning goals — it consisted entirely of preconception-related codes.

In our prior work, we described our observations about what learning goals fell into which SOLO stage. In this paper, we will focus on the clusters that corresponded to the transitions between SOLO stages — the barriers that students faced in development, and the tactics that instructors used to facilitate student development.

We will also describe the codes which did not neatly fit into our SOLO-inspired organization of codes: the difficulties students faced surrounding math and programming, as well as the instructional tactics which spanned multiple SOLO levels.

4 PRIOR WORK: LEARNING GOALS CLASSIFIED BY THE SOLO TAXONOMY

While this paper focuses on the PCK needed to transition students from one SOLO stage from another, it is first worth describing the learning goals that we classified into each SOLO stage.

The SOLO taxonomy has been found useful for classifying learning objectives and designing curricula and course materials [3, 5]. We found that learning goals described as easy to teach were consistently at the lower end of the SOLO taxonomy, while harder goals were found consistently at the higher end. Table 2 provides a broad overview of the learning goals found in our data.

Table 2: Instructor-Based Taxonomy of ML Learning Goals for Non-Majors using SOLO Stages

Unistructural Stage
(Describe, Trace, Interpret, Identify) simpler ML algorithms that do not require a high-level of prerequisite math knowledge.
Multistructural Stage
Compute a baseline.
Implement an algorithm with black box.
Plan for training and testing phase by splitting data.
Convert raw data into computer input.
Relational Stage
Relating between real life and computer input/output.
Evaluate performance with consideration to the bias-variance tradeoff.
Apply evaluation to tune parameters to improve performance.
Extended Abstract Stage
Develop a model from scratch to solve a problem.
Communicate performance and limitations of model.

After establishing what students learn at each SOLO stage, we turned to PCK for supporting development. We will describe preconceptions students have (which affects the transition to the unistructural stage) and then the student barriers and pedagogical tactics corresponding to the transitions between the other SOLO stages.

5 PRECONCEPTIONS

Preconceptions are an important element of PCK. It is important for educators to know where students are beginning from, particularly to support them reaching a unistructural stage of learning.

When asked for student preconceptions, P10 responded that students have asked, "Can you teach me to machine learn?" It is important to note the marked difference in the ML exposure that non-majors may have vs. those with a CS/math background, seen in something as simple as using the term "machine learning" correctly. In the interviews, we found three recurrent preconceptions:

- **ML is Important:** students believe that ML is extremely powerful and conclude that it is the "next big thing" and will "get you a job". According to participants, this reflects a common student motive for taking an ML course. P10 spoke to this saying that research scientists, as students, express a need to use ML because more publications are coming out using ML techniques.
- **Heard of ML from Popular Media:** Students have heard of ML technology through popular media, such as news articles on AlphaGo and autonomous cars. P9 offered repercussions of this; students read sensationalized media on ML and they[1] have to repeatedly debunk the limits of ML during the duration of the course.
- **Implementing ML is not accessible:** Some participants noted their students often feel that they are unfit to implement ML, perceiving that they need a strong CS/math background. This has been debunked in the literature [11, 35].

These preconceptions concern themselves with the reputation of ML rather than how ML works. This reputation can cause students to view ML as an intimidating subject to learn and/or to overestimate ML's abilities.

6 FACILITATING STUDENT DEVELOPMENT ALONG THE SOLO STAGES

The barriers that students face in learning ML change as the students develop in their understanding of the topic. Anticipating and appreciating student difficulties is an important component of PCK, as is having instructional tactics for these various barriers.

Here we describe the different barriers students face as they progress along the SOLO taxonomy, alongside the tactics reported by participants that were targeted at particular SOLO stages.

6.1 Getting Students to a *Unistructural Stage*

In the SOLO taxonomy, students first begin at a prestructural stage where they do not understand the topic at hand. This corresponds to the preconceptions we discussed in Section 5. The prestructural stage is aptly described by P3 as a *lack* of understanding rather than *mis*understanding.

The next stage of the SOLO taxonomy is the unistructural stage, wherein students can understand a single element of the topic. What PCK do ML educators have about facilitating the transition from prestructural to unistructural understanding of ML?

Participants noted two tactics for facilitating this transition, and one barrier that students encounter.

[1]We refer to all participants with gender neutral pronouns throughout this paper.

6.1.1 Barrier: Human Thinking vs. Computer Processing. It was mentioned frequently that students attribute human qualities to computers, for example, that computers have intuition or common sense that will influence their decision. Or that computers can take in data (e.g., language, images) and process it similar to humans. Psychology students specifically want to map ML to the the brain and models of learning previously encountered during their psychology studies. This barrier is possibly due to a lack of exposure to algorithmic thinking, considering their non-major background. P8, who teaches to both CS and non-majors, explicitly noted that this was an issue with only the non-major students.

6.1.2 Tactic: Working through Simple Problems by Hand. Participants described working through simple problems by hand to promote student understanding of ML algorithms. The key to these problems is that they are low dimensional, so students can trace the steps of the algorithm sequentially.

6.1.3 Tactic: Simulating an Algorithm. Along the same line of understanding algorithms, participants promoted simulation of algorithms, which can come in different forms. Two participants described activities where their class simulates an algorithm as a whole. For example, P3 turns the class into a multi-level perceptron where every student acts as a function within the network. Students pass a note as input and output, following the network structure, with each student altering the note to their function. The goal is to demystify the black box by having students experience an algorithm's sequential logic. Other forms include interactive visualizations of algorithms (e.g., pyLDAvis [2] for Topic Modeling).

6.2 Getting Students to a *Multistructural* Stage

When at a multistructural stage, students learn the individual steps needed to transform an algorithm into model (e.g., data cleaning, creating features, using Sckit-learn), but cannot yet see how the individual steps all come together.

In the context of machine learning, we define an *algorithm* to be a set of instructions. A *model* is the result of training data being inputted into an algorithm to make predictions on unseen data.

There are no overarching themes in the data for transitioning to the multistructural stage. Participants had comments for this stage that were not expressed by others. This indicates future work needed to further investigate these two stages and their progression.

6.3 Getting Students to a *Relational* Stage

At a relational stage, students can combine their skills learned in previous stages to build and tune a model, as well as students making design decisions about features and parameters. Participants noted these were difficult for students to learn.

6.3.1 Barrier: Underestimating Decision-Making in ML. According to participants, students believe that ML requires little human decision. It is more difficult to design a model (i.e., choosing algorithm, features, and parameters) than to implement a model. P6 described that students were surprised that humans are responsible for deciding when to stop tuning a model, expecting more automation and less decision-making.

[2]Available at https://github.com/bmabey/pyLDAvis

6.3.2 Tactic: Strategically Choosing Data Set(s). Participants reported using low-dimensional data sets throughout a course, whether in assignments or in-class examples. Students can grasp more easily what is happening to the input and analyze the output without getting lost in a large number of features. For example, P6 described using the IRIS data set[3] because of its low-dimension (four attributes). Even with its low-dimensionality, effective models can be trained on it. There are other benefits to low-dimensions, including faster run time and less data cleaning.

In addition, participants described minimizing the data sets they use throughout the courses. As P10 stressed, every new data set is a cognitive barrier for students because students need to get acquainted with its unique format and in addition, will probably need to clean it, which takes time. P6 uses the IRIS data set for every new algorithm they introduce. By using it on each algorithm taught, they aim to show that the same question can be addressed using different tools. They concluded that it helps students build confidence, noting that students often try their unrelated final project models on the IRIS data set as a reference point.

6.4 Getting Students to an *Extended Abstract* Stage

Students at a extended abstract stage are able to develop a model to solve a specific problem using their model building skills from the previous stage to compare different models and select the best solution. At this stage, students also evaluate whether ML methods are suitable for a problem. Participants noted one barrier and tactic in facilitating student development to this stage.

6.4.1 Barrier: Perceiving the Limits of ML Applications. When given a problem, students lack the immediate inclination to identify the constraints of applying ML as a solution. This may be related to students' over-estimation of the capabilities of ML. P9 says that they have to re-iterate that a model is only as good as the data, i.e., ML applications depend heavily on quality and quantity of data.

6.4.2 Tactic: Open-Ended Problems. Participants described teaching their students through open-ended problems for assignments, class activities, and final projects, which would involve developing a new model from scratch.

7 GENERAL PEDAGOGICAL TACTICS

Whereas the tactics from Section 6 were targeted to particular learning goals or transitions, participants additionally described tactics that they used generally. By "generally" we mean they used this tactic for numerous transitions, or they mentioned the tactic without a reference to a specific learning goal. These were generally used throughout a course, and are sufficiently flexible that they can be used for a variety of learning goals.

7.1 Visualization

When answering Question 3 or 6 (see Appendix A), participants overwhelmingly mentioned "visualization" broadly. This term encapsulates a wide range, including some of the tactics that were presented in previous sections. In response to Question 3, participants expressed that visualized concepts are easier. This could be because a concept lends itself to visualization (e.g., could be drawn on a blackboard) or because a concept has an existing visualization (e.g., online interactive demos). This was especially cited for explaining algorithms.

There was no consensus on a type of visualization; each participants had their own methods. One participant who did not include programming, advocated for drag-and-drop type interfaces for building ML processes, such as Microsoft Azure Machine Learning Studio. Another created demos on Jupyter notebooks that his class could follow along to. Finally, a few mentioned data plotting.

7.2 Relating to the Real World

Using real world applications was strongly promoted by participants. This can come in the form of showing how certain models solve common problems, such as Naive Bayes for effective spam filtering. Or, students solving socially significant issues using public data sets (e.g., predicting benign or malignant tumors using the Breast Cancer Wisconsin Data Set[4]).

It may not even have to do with solving a problem, but the context. Two participants described using data related to their school; school newspaper archives and the course catalog as a subject of topic modeling and NLP respectively. As a final example of diverse applications, one participant has students read O'Neil's *Weapons of Math Destruction* [27], which concerns itself more with how ML causes problems within society. This real world relation contextualizes ML's importance by highlighting to students its increasing relevance in solving and, even, creating contemporary problems.

7.3 Domain Specificity

The majority of our participants teach ML to a specific discipline, to which they expressed the importance of teaching to the discipline of the students. P6's final comment on teaching non-majors was that having an instructor from the students' domain is more effective than sending students to the CS department. P9 agreed with this saying if an instructor approaches teaching ML to non-majors as a CS instructor, they will lose students in the math and programming.

Being domain specific helps with student motivation. P2 noted that their students are "eager to finally combine psychology and computation". In addition, P7 made an interesting comment in answering question 4 that student difficulty depends on disciplinary norms, citing difference in math and programming exposure across disciplines. This may be another reason to support domain-specificity.

According to the participants' answers, being domain specific can look like focusing course content to relevant ML methods to the students' field, showing domain-specific examples, or giving domain-specific problems or data sets to solve.

8 GENERAL STUDENT BARRIERS

Math and programming arose as difficult topics for students. These topics were broadly mentioned and our interview questions did not aim to capture their details. From the participants' comments, student difficulty with math and programming is independent from learning ML, yet necessary to address in order to teach ML.

[3] Available at https://archive.ics.uci.edu/ml/datasets/iris

[4] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

8.1 Math as a Barrier for Students

Math was consistently brought up by participants, with probability and linear algebra noted as subtopics. P2 sums up the participants' thoughts on math when they say "anything mathematical seems to elicit some concern in many students". P7 describes their students as "self-identified as not able to do math". This reflects that a source of difficulty is that math itself causes anxiety, rather than its application. Additionally, there were comments on the lack of prerequisite(s) in math or retainment of prerequisite(s) math course(s) as a source of the difficulty.

Some instructors actively avoided teaching with math as a way around this barrier. As much as there were comments on math difficulties, only one participant offered a way to deal with math difficulties that did not require omitting math. P7 purposefully speaks of students as "not being good at math yet" and empowers their journalism students by saying that because they are good at learning, they will be able to learn the necessary math concepts.

8.2 Programming as a Barrier for Students

Programming did not come up as consistently as math, but not all participants include programming in their courses. Non-major students usually have at most one programming course done, which is one of the sources of student difficulty. This is also a reason why some participants do not include programming and opt to use visual interfaces for implementing ML (e.g., Microsoft Azure).

Although programming was broadly mentioned, the most agreed difficulty on programming was debugging. P3 had an interesting comment that "if students had a bug, they saw it more as a problem with programming than understanding of ML concepts so they don't get demotivated", which may be a helpful re-framing.

9 DISCUSSION

Through our interviews with ten participants, we uncovered a range of themes that begin to explore ML PCK for non-majors. We suspect that some of the results presented here are not specific to non-majors, but may be transferable to CS and statistics students. For example, there is overlap between our results and Lavesson's [21] course evaluation findings on easier and more difficult topics in ML (see Section 2.2). This is especially notable because Lavesson presents the perspectives of CS students.

9.1 Limitations

Our short interview format helped us recruit more participants and gain breadth. However, as the interviews lasted on average of 21 minutes to accommodate our uncompensated participants' busy schedules, this affected the depth of conversation. We did not interview any students nor observe any classrooms, thus limiting our study to the perspectives of instructors.

There were considerable differences in the ML course content taught by our participants and the motivation of their students for learning the subject. While this helps survey a broad scope of course content, it may have reduced consensus on the themes since the participants were often describing different topics and student populations, unique to their respective courses. However, given that ML is a new subject to be taught to non-majors, these issues arise naturally with the subject of study.

9.2 Future Work

To the best of the authors' knowledge, this is the first work on ML PCK. We hope to provide a foundation for future PCK studies in ML. Our interview format did not focus on specific ML concepts because we did not know which concepts, across disciplines, would be consistently relevant. Having identified several learning goals as well as common barriers, more specific questions can be developed to capture more detailed PCK.

Although "visualization" was a frequently mentioned tactic, participants lacked specificity as to what it meant. There is opportunity to more clearly define what constitutes visualization of ML and best practices for doing so.

More work is needed on the role of math and programming while learning ML. How do these subjects facilitate understanding while teaching/learning ML? Is math development and programming development separate from ML development? And in terms of course design for non-majors with little prerequisites: how much math needs to be included? Is programming necessary?

10 CONCLUSION

We identified examples of pedagogical content knowledge (PCK) for teaching machine learning, namely in the forms of preconceptions and barriers faced by students along with instructional tactics.

Student preconceptions include ideas that ML is important, but also not accessible. The data from our participants are evidence that it is possible to teach ML to those with little to no math/CS background. Instructors reported students having difficulty appreciating the human decision-making aspects of ML, and overestimating the power of ML to solve real-world problems. Visualization was a popular tactic for teaching ML; it appears that more visualization tools would be useful for ML educators.

PCK is important to building a teacher's competence. Our results highlight opportunities for innovation in teaching ML as well as the relationships ML has with math and programming.

A INTERVIEW QUESTIONS

(1) Could you briefly describe the context in which you teach machine learning? (i.e. what background do your students have and what are the motivations?)
(2) What are some common ML preconceptions that students have before taking your course?
(3) What ML concepts do students have the easiest time with?
(4) What ML concepts do students have most difficulty grasping?
(5) What are the most common mistakes students make?
(6) Are there any examples, analogies, or other methods of explaining concepts that you find particularly effective?
(7) Any other comments on teaching non-technical students machine learning?
(8) Do you have any formal education in teaching? If so, when?

ACKNOWLEDGMENTS

We thank the participants for their time and insights. We also acknowledge Horatiu Halmaghi and Jessica Quynh Tran for partaking in the qualitative analysis. This work was supported with funding from NSERC.

REFERENCES

[1] Deborah Loewenberg Ball, Mark Hoover Thames, and Geoffrey Phelps. 2008. Content Knowledge for Teaching: What Makes It Special? *Journal of Teacher Education* 59, 5 (2008), 389–407. https://doi.org/10.1177/0022487108324554 arXiv:https://doi.org/10.1177/0022487108324554

[2] John B Biggs and Kevin F Collis. 1982. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.

[3] Gillian M Boulton-Lewis. 1995. The SOLO taxonomy as a means of shaping and assessing learning in higher education. *Higher Education Research and Development* 14, 2 (1995), 143–154.

[4] Danah Boyd and Kate Crawford. 2012. Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, communication & society* 15, 5 (2012), 662–679.

[5] Claus Brabrand and Bettina Dahl. 2009. Using the SOLO taxonomy to analyze competence progression of university science curricula. *Higher Education* 58, 4 (2009), 531–549.

[6] Erik Brynjolfsson and Andrew McAfee. 2012. *Race against the machine: How the digital revolution is accelerating innovation, driving productivity, and irreversibly transforming employment and the economy*. Brynjolfsson and McAfee.

[7] Michael Clancy. 2004. Misconceptions and attitudes that interfere with learning to program. *Computer science education research* (2004), 85–100.

[8] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. 2012. Detecting and understanding students' misconceptions related to algorithms and data structures. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 21–26.

[9] Virginia Eubanks. 2018. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.

[10] Carl Benedikt Frey and Michael A Osborne. 2017. The future of employment: how susceptible are jobs to computerisation? *Technological forecasting and social change* 114 (2017), 254–280.

[11] Yolanda Gil. 2016. Teaching Big Data Analytics Skills with Intelligent Workflow Systems.. In *AAAI*. 3997–4088.

[12] Randall E Groth. 2007. Toward a conceptualization of statistical knowledge for teaching. *Journal for research in Mathematics Education* (2007), 427–437.

[13] Michael Guerzhoy and Renjie Liao. 2018. Understanding How Recurrent Neural Networks Model Text. http://modelai.gettysburg.edu/2018/rnntext/index.html

[14] Heather C Hill and Deborah Loewenberg Ball. 2004. Learning mathematics for teaching: Results from California's mathematics professional development institutes. *Journal for research in mathematics education* (2004), 330–351.

[15] Heather C Hill, Brian Rowan, and Deborah Loewenberg Ball. 2005. Effects of teachers' mathematical knowledge for teaching on student achievement. *American educational research journal* 42, 2 (2005), 371–406.

[16] Heather C Hill, Stephen G Schilling, and Deborah Loewenberg Ball. 2004. Developing measures of teachers' mathematics knowledge for teaching. *The elementary school journal* 105, 1 (2004), 11–30.

[17] Aleata Hubbard. 2018. Pedagogical content knowledge in computing education: a review of the research literature. *Computer Science Education* (2018), 1–19.

[18] Peter Hubwieser, Marc Berges, Johannes Magenheim, Niclas Schaper, Kathrin Bröker, Melanie Margaritis, Sigrid Schubert, and Laura Ohrndorf. 2013. Pedagogical content knowledge for computer science in German teacher education curricula. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*. ACM, 95–103.

[19] Lisa C Kaczmarczyk, Elizabeth R Petrick, J Philip East, and Geoffrey L Herman. 2010. Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, 107–111.

[20] Dona M. Kagan. 1990. Ways of Evaluating Teacher Cognition: Inferences Concerning the Goldilocks Principle. *Review of Educational Research* 60, 3 (1990), 419–469. https://doi.org/10.3102/00346543060003419 arXiv:https://doi.org/10.3102/00346543060003419

[21] Niklas Lavesson. 2010. Learning machine learning: a case study. *IEEE Transactions on Education* 53, 4 (2010), 672–676.

[22] Hollylynne S Lee and Karen F Hollebrands. 2011. Characterising and developing teachersâĂŹ knowledge for teaching statistics with technology. In *Teaching statistics in school mathematics-challenges for teaching and teacher education*. Springer, 359–369.

[23] John Loughran, Pamela Mulhall, and Amanda Berry. 2004. In search of pedagogical content knowledge in science: Developing ways of articulating and documenting professional practice. *Journal of research in science teaching* 41, 4 (2004), 370–391.

[24] Melanie Margaritis, Johannes Magenheim, Peter Hubwieser, Marc Berges, Laura Ohrndorf, and Sigrid Schubert. 2015. Development of a competency model for computer science teachers at secondary school level. In *2015 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 211–220.

[25] Safiya Umoja Noble. 2018. *Algorithms of Oppression: How search engines reinforce racism*. NYU Press.

[26] Cathy O'Neil. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Publishing Group, New York, NY, USA.

[27] Cathy O'Neil. 2016. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books.

[28] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814* (2016).

[29] Elizabeth Patitsas, Michelle Craig, and Steve Easterbrook. 2013. On the countably many misconceptions about #hashtables. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 739–739.

[30] Yizhou Qian and James Lehman. 2017. StudentsâĂŹ misconceptions and other difficulties in introductory programming: a literature review. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1.

[31] Mara Saeli, Jacob Perrenet, Wim M. G. Jochems, and Bert Zwaneveld. 2011. Programming in Secondary School : A Pedagogical Content Knowledge Perspective.

[32] Otto Seppälä, Lauri Malmi, and Ari Korhonen. 2006. Observations on student misconceptions: A case study of the Build–Heap Algorithm. *Computer Science Education* 16, 3 (2006), 241–255.

[33] Lee S. Shulman. 1986. Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher* 15, 2 (1986), 4–14. https://doi.org/10.3102/0013189X015002004 arXiv:https://doi.org/10.3102/0013189X015002004

[34] Douglas Turnbull. 2012. Music Genre Classification. http://modelai.gettysburg.edu/2012/music/

[35] Thomas Way, Lillian Cassel, Paula Matuszek, Mary-Angela Papalaskari, Divya Bonagiri, and Aravinda Gaddam. 2016. Broader and Earlier Access to Machine Learning. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 362–362.

[36] Thomas Way, Mary-Angela Papalaskari, Lillian Cassel, Paula Matuszek, Carol Weiss, and Yamini Praveena Tella. 2017. Machine Learning Modules for All Disciplines. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 84–85.

[37] Aman Yadav, Marc Berges, Phil Sands, and Jon Good. 2016. Measuring computer science pedagogical content knowledge: An exploratory analysis of teaching vignettes to measure teacher knowledge. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. ACM, 92–95.