What Is Hard About Teaching Machine Learning to Non-Majors? Insights From Classifying Instructors' Learning Goals

ELISABETH SULMONT, McGill University, Canada
ELIZABETH PATITSAS, McGill University, Canada
JEREMY R. COOPERSTOCK, McGill University, Canada

Given its societal impacts and applications to numerous fields, machine learning (ML) is an important topic to understand for many students outside of computer science and statistics. However, machine learning education research is nascent, and research on this subject for non-majors thus far has only focused on curricula and courseware. We interviewed ten instructors of ML courses for non-majors, inquiring as to what their students find both easy and difficult about machine learning. While ML has a reputation for having algorithms that are difficult to understand, in practice our participating instructors reported that it was not the algorithms that were difficult to teach, but the higher-level design decisions. We found that the learning goals that participants described as hard to teach were consistent with higher levels of the Structure of Observed Learning Outcomes (SOLO) taxonomy, such as making design decisions and comparing/contrasting models. We also found that the learning goals that were described as easy to teach, such as following the steps of particular algorithms, were consistent with the lower levels of the SOLO taxonomy. Realizing that higher-SOLO learning goals are more difficult to teach is useful for informing course design, public outreach, and the design of educational tools for teaching ML.

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

Additional Key Words and Phrases: Machine learning; computer science education

ACM Reference Format:
Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R. Cooperstock. 2019. What Is Hard About Teaching Machine Learning to Non-Majors? Insights From Classifying Instructors' Learning Goals. 1, 1 (August 2019), 17 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Machine learning (ML) has become an important technology in today's society, prompting a need for an audience broader than the traditional computer science (CS) and statistics communities to learn about it. As a result, more research is needed to understand and improve how we teach machine learning to these non-traditional audiences.

Authors' addresses: Elisabeth Sulmont, McGill University, School of Computer Science, Montreal, QC, Canada, elisabeth.sulmont@mail.mcgill.ca; Elizabeth Patitsas, McGill University, Department of Integrated Studies in Education and the School of Computer Science, Montreal, QC, Canada, elizabeth.patitsas@mcgill.ca; Jeremy R. Cooperstock, McGill University, Department of Electrical and Computer Engineering, Montreal, QC, Canada, jer@cim.mcgill.ca.

XXXX-XXXX/2019/8-ART $15.00
https://doi.org/0000001.0000001

As machine learning is a relatively new field, machine learning education is in its early days, with the majority of the existing work targeting CS students. Our goal in this work 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 and ML education to develop. This paper is part of a larger project aimed at doing exploratory research on how to teach machine learning to non-majors. In the context of this paper, we define "non-majors" as students without a background in CS or statistics.

With an understanding of machine learning, non-majors could better critique ML and its implications to society, such as job automation, privacy concerns, and algorithmic bias. This is of obvious relevance to journalists, politicians, and activists. As an additional benefit, equipping non-majors with ML skills to combine with their domain knowledge will improve current ML applications and help develop new applications across domains.

Vital to equipping these non-majors with ML skills are the educators who will teach them. These educators will need not only ML knowledge but an ability to communicate and work with people outside computer science and statistics. Since we were interested in supporting the educators, we identified *pedagogical content knowledge* (PCK) as a useful construct for our study. Shulman [27] introduced PCK as the knowledge needed by an educator to teach a particular subject matter and it has since been viewed as an important component of effective pedagogy in education research. 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.

Our approach employed interviews of ten instructors with experience teaching machine learning to non-majors. Since the participants were instructors, rather than the students themselves, our questions were focused on what instructors found easy and difficult to teach. In our qualitative analysis of the resulting interview data, we not only identified PCK but also learning goals, the latter of which is the focus of this paper. In previous work [29], we present the interview data related to PCK.

In our qualitative analysis, we found the Structure of Observed Learning Outcomes (SOLO) stages to be a useful organizing approach for analyzing the data collected, in particular, regarding participants' learning goals and their perceptions of difficulties faced by students. In this paper, learning goals described as easy were consistently at the lower end of SOLO, while harder goals were found consistently at the higher end. Our present findings suggests that participants find it more difficult to teach high-level design decisions, rather than particular algorithms.

1.1 Structure of the Observed Learning Outcome (SOLO) Taxonomy

Biggs and Collis' Structure of the Observed Learning Outcome (SOLO) taxonomy [4] is a framework to classify increasingly complex learning outcomes. Its development was motivated by a desire to improve how student learning is measured, with an emphasis on qualitative rather than quantitative evaluation, and it has been used to better evaluate students and design curricula with appropriate learning objectives [3, 6, 8].

The SOLO taxonomy is made up of five stages, as described in Table 1: prestructual, unistructural, multistructural, relational, and extended abstract. As competence of a given subject increases, students are able to take individual aspects and integrate them for more complex tasks such as invention and prediction. Of note is that SOLO does not represent a student's understanding of a course, but of a topic, meaning that a student can be at different stages for topics within a course.

The SOLO taxonomy has been used fruitfully in CS [9, 13, 19, 22, 25, 26, 35] and in mathematics education [1, 10, 20, 23, 32] to track and describe student understanding of specific topics.

Stage	Description	Verbs for Intended Learning Outcomes
Prestructural	lack of understanding	*n/a*
Unistructural	one aspect	*memorize, identify, recognize, count, define, draw, find, label, match, name, quote, recall, recite, order, tell, write, imitate*
Multistructural	several independent aspects	*classify, describe, list, report, discuss, illustrate, select, narrate, compute, sequence, outline, separate*
Relational	integrating aspects into a system	*apply, integrate, analyse, explain, predict, conclude, review, argue, transfer, make a plan, compare, contrast, organize, debate, construct, paraphrase, solve a problem*
Extended Abstract	apply aspects to beyond what has been given	*theorize, hypothesize, generalize, reflect, generate, create, compose, invent*

Table 1. SOLO Taxonomy Stages [2]

For example, Lister et al. [22] applied the SOLO taxonomy on responses to code-reading exercises. Castro and Fisler [9] developed a SOLO taxonomy for different program design skills after tracking student progress in a full CS1 course. Sheard et al. [25] assessed novice programmers using the SOLO taxonomy by analyzing responses to exam questions.

1.2 Classifying Learning Objectives in Computer Science

Learning taxonomies, such as SOLO, have not only been used to track and assess student understanding in CS, but also as a tool to create and analyze learning objectives [11].

Using the SOLO taxonomy, Brabrand and Dahl [7] compared intended learning outcomes (ILOs) of 550 courses across science subjects from two Danish universities. These ILOs were originally formulated with consideration to the SOLO taxonomy. Braband and Dahl analyzed the verbs used in the ILOs by mapping them to different SOLO levels. The authors found that relative to natural science subjects, ILOs from mathematics use lower SOLO stages, while those from CS use higher ones. They noted that CS stands out with the majority of the competences being qualitative (e.g., programming-related), whereas the majority of mathematics and natural science competences are quantitative. They state, "[a]lthough the SOLO taxonomy might not fit each department equally well, it is [sic] good tool to help create a discussion about the purposes of a course and the formulation of ILOs" [7, 12].

The SOLO taxonomy comes out of the same research tradition of constructivism as other learning taxonomies such as Bloom's taxonomy [5]. Given the parallels between different learning taxonomies and their common theoretical basis, we found it informative to refer to literature in CS education using other constructivist learning taxonomies.

In CS, Bloom's taxonomy is most widely used for classifying learning objectives [11]. Its main difference from the SOLO taxonomy, as described by Biggs and Collis, is that "[t]he Bloom Taxonomy is really intended to guide the selection of items for a test rather than to evaluate the quality of a student's response to a particular item" [4, 13].

Oliver et al. [24] conducted an analysis of six courses of varying levels from an IT degree program in an Australian university. The researchers rated the assessment requirements (e.g., assignments and exams) in each course, according to Bloom's taxonomy. For example, the first level of Bloom's taxonomy relates to the ability to recall facts, while the sixth level relates to making judgments [5]. Their analysis found that the lower-level courses were comparatively high on Bloom's taxonomy, whereas the higher-level courses were rated lower. They hypothesize a reason for this result could be that "the necessity of covering a high quantity of difficult technical material tends to result in insufficient attention being given to higher-level tasks" [24, 5].

Starr et al. [28] present a process for applying Bloom's taxonomy to specify and refine assessable learning objectives in CS courses. The authors describe positive results: "[o]nce Bloom levels have been applied to learning objectives, the instructor's job becomes much easier, in that the problem of designing a lecture to cover a particular topic becomes less nebulous, more clearly specified. This also applies to in-class activities, homework assignments, and test questions" [28, 264].

Szabo and Falkner [30] describe how Neo-Piagetian stages can be used as a tool to help develop a coherent curriculum. Specifically, they demonstrate how Neo-Piagetian theory can be used to identify any overlooked prerequisite concepts and leaps where students are assessed at a higher level than taught. An application of this framework is presented in an analysis of three consecutive first year programming courses.

Gluga et al. [14, 15] investigated whether CS educators could learn and reliably apply Bloom's taxonomy and Neo-Piagetian stages to classify programming course material. Their findings demonstrated that educators could effectively and confidently classify assessment materials after an hour-long tutorial on the respective taxonomies within a programming context.

For an alternate framework, Guzdial [17] has speculated that the Structure-Behaviour-Function (SBF) Theory could be used to understand computer science learning; however this has not yet been explored in the literature. SBF is oriented around the understanding of complex systems and the relationships between aspects of a system [18].

1.3 Teaching Machine Learning

Education research on machine learning education remains nascent, in large part because machine learning is a relatively new subject in university curricula. What little literature exists on this topic can be found predominantly in publications from the Symposium on Educational Advances in Artificial Intelligence, which first met in 2010. This includes assignment designs centered on machine learning, such as classifying songs by genre [31] and generating "fake English" text from a recurrent neural network [16].

Of relevance to our study is Lavesson's [21] case study of a master's level machine learning course (30 students) to see how prior knowledge of prerequisites (mathematics, statistics, and programming) affected student achievement of the course's learning objectives. There were four stages of data collection: self-assessment of prerequisites, diagnostic test of prerequisites, course results, and course evaluations.

Eleven students participated in the self-assessment stage and the diagnostic tests. Students rated their knowledge of statistics the lowest (slightly above average level of understanding), which correlated with their diagnostic tests. The only significant difference between the two tests was that students felt confident with linear algebra while their diagnostic test results for that section were comparatively poor. With only eight participants for the course results phase, Lavesson states that the data was too limited for statistical testing of correlation between the first three stages.

In the anonymous student 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 algorithms empirically with statistical tests and experiments. The implementation of ML algorithms using programming was found to be easiest out of all learning outcomes.

1.3.1 Teaching Machine Learning to Non-Majors. The literature on teaching machine learning to non-majors is much more limited than the already minimal literature on machine learning education. Thus far, this literature focuses on curricula for non-majors, as described below.

We were therefore motivated to widen the literature on ML education for non-majors, beyond papers on course materials. For machine learning education to develop, we need to understand how and why non-majors learn machine learning and how to teach to these students specifically.

Way et al. [33] [34] presented their findings from a multi-year project called "Broader and Earlier Access to Machine Learning." The project's three goals were to "(1) identify machine learning topics and relevant presentation techniques for undergraduates and faculty across disciplines, (2) produce stand-alone modules that can be adapted for learning in a variety of fields, and (3) disseminate the knowledge to a wide audience" [33, 362], which resulted in an online repository of flexible module materials[1] for non-major student learning. Way et al. [34] evaluated their modules through pre- and post-tests and found that students gain and retain knowledge.

Gil [12] designed a data science course with machine learning topics for non-programmers, through a workflow paradigm and a visual interface called WINGS. WINGS is an intelligent semantic system that allows users to create, tune, and compare machine learning processes. It checks that data fits semantic constraints and allows students to run high level methods on large real-world data without programming. Pre-tested on four non-majors, Gil proposes that learning the basic concepts of data science without programming is more approachable due to less time investment (e.g., required courses in programming, lessons spent on programming). Gil concluded that students are given a concrete basis to understand AI concepts through the WINGS-based course.

2 METHODS

We began this study as an exploratory investigation into machine learning pedagogical content knowledge (PCK) [27] for non-majors. Our question design was focused around PCK, and we interviewed teachers who have taught to non-majors as a way to understand the breadth of what machine learning PCK could be. As we conducted our qualitative analysis, a pattern emerged that it was harder for participants to teach learning goals that are consistent with being higher in the SOLO taxonomy, and easier to teach learnnig goals that are consistent with being lower in the SOLO taxonomy. So while we did not enter the study expecting to use SOLO, as is common with exploratory research, its suitability emerged through our analysis.

2.1 Participants

Participants were recruited through cold emails and mailing lists related to computer science and data science education. The only requirement for participation was experience teaching machine learning concepts to non-majors. Table 2 provides an overview of the ten participants and their setting for teaching machine learning. We can separate courses into two broad types, dedicated introductory courses in machine learning for non-majors leaning towards professional development (P4, P9, P10) and discipline specific courses with a component on machine learning (the remaining participants). All are professors, except for three instructors (P4, P9, P10) who all have PhDs

[1]Available at http://computingportal.org/MachineLearning.

and, perhaps not incidentally, teach courses leaning towards professional development. All of the participants' institutions are located in Canada or the USA, with the exception of P5 (Europe).

It is important to note upfront that interviewees have diverse learning outcomes for their students. This is affected by factors such as duration of machine learning lesson plans, type of setting (university, professional, etc.), background of students, and how much math and programming were included. For example, the goal of some courses is to get students to implement machine learning, while for others, it is for students to gain a high enough level understanding to discuss social implications. In addition, there are different levels of duration and detail in the machine learning sub-topics covered in each of the participants' courses.

Table 2. Interview Participants

ID	Position	Setting	Teaching
P1	Information & Library Science 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	Introductory machine learning for professional data science master program
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 & Computer Science Professor	Private Liberal Arts College	Topics in AI
P9	Data Scientist & Instructor	AI Research Center	Workshops for upper-level business people
P10	Data Scientist & Instructor	Online Learning Platform	Machine learning courses for research scientists, mainly from life sciences

2.2 Structured Interviews

The development of the interview questions was informed by literature on pedagogical content knowledge. The questions were therefore designed to probe for difficulties, misconceptions, and teaching strategies. Table 3 lists the questions we used for our interviews. The questions did not focus on specific topics within machine learning because we could not ensure all teachers chose to cover the same topics, especially across different disciplines.

All interviews were conducted over voice-call, except for Participant #2 (P2) who gave their answers over written 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.

#	Question
1	Could you briefly describe the context in which you teach machine learning (ML)? (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?

Table 3. Structured Interview Questions

2.3 Approach

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

3 QUALITATIVE ANALYSIS

3.1 Initial coding

First, all four coders read the transcripts in their entirety and based on first impressions we created four initial categories to organize the codes. These categories were student preconceptions, easy for students, difficult for students, and tactics. As we started coding we added two more categories: hard to teach, and teacher goals. For each category we used a different colour of post-it note. We free coded the transcripts in order of interview questions (i.e., code all the transcripts for question 2, then code all the transcripts for question 3, etc.). The initial codes were loosely grouped by our initial categories. We wrote each code onto a post-it note of the appropriate colour, and posted the post-it/code on a blank wall.

3.2 Grouping codes

In our next stage of analysis we clustered the codes that were repetitions of the same concept or very close to being repetitions. We then formed groups of codes that naturally fit together, regardless of the category under which they were originally coded. This resulted in 38 groups. Groups titled "Math," "Visual," "Human vs Computer," and "Related to Domain" were the most visibly dominant groups. "Math" and "Programming" were unique groups that encapsulated difficulties from their respective disciplines and did not exclusively occur while learning machine learning (e.g., debugging). In addition, there were groups related to pedagogy (tools, techniques) and general ML preconceptions.

In examining our groupings, we noticed that the group titled "Design decisions" was mostly full of pink ("difficult for students") post-it notes, whereas the group on teaching students a given ML algorithm was dominated by the green ("easy for students") post-it notes.

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 lower SOLO/Bloom/SBF stages, the hard to teach learning goals were grouped along issues that correspond to higher levels of the SOLO/Bloom/SBF stages, and groups related to the middle stages of SOLO/Bloom/SBF had a mix of pink and green. With this realization, we wanted to re-cluster the codes using a learning taxonomy to see if it provided any interesting insights.

3.3 Choosing SOLO

We decided on the SOLO taxonomy over other frameworks to re-cluster the codes because we believe it to be the most insightful in describing student understanding of machine learning. We specifically considered Bloom's taxonomy [5] and SBF theory [18] as options. Although the former is more widely used, it does not share the same empirical basis as SOLO. Its popularity can be attributed to its earlier introduction into education research. We also did not find as neat a mapping from our codes to the SBF framework as we did for the SOLO taxonomy, and SBF has not been as widely used in the CS education community. This led us to regroup our codes in a manner that was influenced by the SOLO taxonomy.

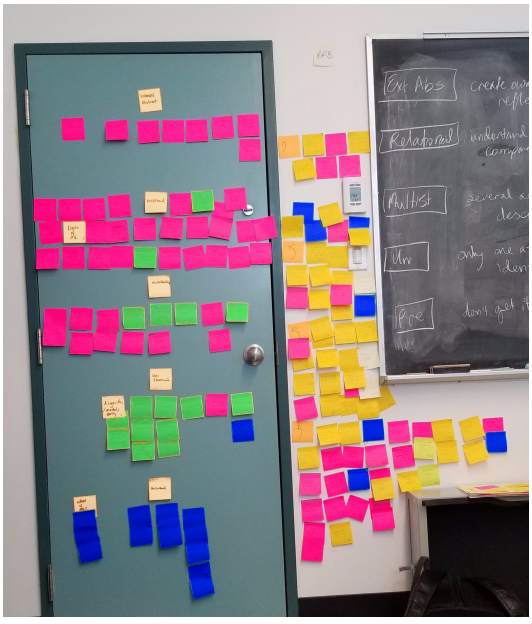

Fig. 1. Regrouping codes based on the SOLO taxonomy. The codes which represent learning goals are on the door, grouped by SOLO stage. Pink is 'hard to teach', green is 'easy to teach', and blue is 'preconceptions'. The prestructural group is entirely full of blue post-its, the unistructural group is overwhelmingly green, multistructural is a mix, and both relational is mostly pink, and extended abstract is entirely pink.

3.4 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, and one for each transition between stages (shown in Figure 1). Note that unlike the other SOLO stages, the prestructural stage group did not have learning goal related codes — it consistent entirely of preconception-related codes.

The transitionary clusters (to the right of the door in Figure 1) were characterized by instructional tips and techniques for assisting students transition from one SOLO stage to another. This paper is part of a larger research project; in this paper we focus on how the difficulty of teaching particular learning goals correspond to the SOLO stages; previous work [29] examines the pedagogical content knowledge for transitioning students from one SOLO stage to another.

4 TAXONOMY OF ML LEARNING GOALS FOR NON-MAJORS USING SOLO STAGES

In our interviews, instructors described their students struggling less with the actual algorithms and more with the design decisions around them. Analysis of responses indicated that from the perspective of the instructors, students had an easier time with course material lower on the SOLO taxonomy and a harder time with material higher on the taxonomy.

In this section, we present our resulting classification of machine learning learning goals into the pertinent SOLO stages based on our coding. Table 4 provides a broad overview of the learning goals at each stage, which we elaborate upon in the following sections. The learning goals presented are by no means exhaustive.

Stage	Example Learning Goals for Non-Majors
Unistructural	*(Describe, Trace, Interpret, Identify)* simpler ML algorithms that do not require a high-level of prerequisite math knowledge.
Multistructural	*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	*Relate* 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	*Develop* a model from scratch to solve a problem.
	Communicate performance and limitations of model.

Table 4. Instructor-Based Taxonomy of ML Learning Goals for Non-Majors using SOLO Stages

4.1 Prestructual: Lack of Understanding

As the prestructural stage represents a student's initial lack of competence in a topic, no learning goals were classified within it. Nevertheless, we include the participants' perspectives of their students' initial understanding to provide some baseline and context for the growth in goal complexity.

Commenting on common preconceptions, P3 indicated that what distinguishes their students is their *lack* of understanding rather than *mis*understanding. This aptly fits how our participants perceive their students' understanding at the start of a course.

Instructors described students' preconceptions to be more based on the reputation of machine learning rather than on how machine learning works. Specifically, participants indicated that students are usually familiar with the term, "machine learning," through popular media on topics such as AlphaGo and self-driving cars. P9 commented on this, stating, "News doesn't help, because it makes it seem like the computer has a brain that thinks like us." Additionally, they described students as identifying machine learning as the "next big thing".

Participants reported different cases in which this reputation can intimidate the students, make students over-optimistic about the abilities of ML, or be their motivation for taking an ML course. For example, P3 and P6 both described that their students often feel that they are unfit to implement machine learning, citing that they need a strong CS and math background to do so. P10 commented that research scientists, as students, are often motivated to learn machine learning because of the increasing amount of research using machine learning being published.

4.2 Unistructural Learning Goals: Simpler Machine Learning Algorithms

Our participants noted the relative ease of teaching simple machine learning algorithms. Participants described teaching algorithms that do not require a high-level of prerequisite math knowledge, with learning goals which involve simple tracing and description. We would categorize the learning goals into SOLO's unistructural phase, which is concerned with understanding a single aspect of a topic, without necessarily being able to apply it or combine it with other aspects.

Decision trees, k-nearest neighbors (k-NN), k-means, and linear regression were identified from our analysis of the code grouping as representing the largest consensus of "easy for students" codes. Since the focus of each course, and thus, the algorithms covered by our participants, were not identical, the lack of a strong consensus on a single algorithm is understandable. Nevertheless, the algorithms listed are all on the simpler end of the spectrum when compared to other machine learning algorithms in domains such as reinforcement learning and neural networks.

Some participants opined that these are easier algorithms for students either because the algorithms can be visualized well, or they reflect human logic more so than abstract algorithms such as those related to neural networks. For example, P8 hypothesized that k-means clustering and decision trees were the easiest because "they are the most visual and you can see what's going on". P9 described k-NN as an easy topic because it is what students would do naturally, i.e., look at most similar examples to decide on a class.

Additionally, these may be easier algorithms to teach because they do not require a high-level of prerequisite math knowledge, which is less likely to be evidenced among non-majors than CS majors. Accordingly, some participants purposefully avoid teaching the math behind these algorithms, opting for black box implementations, or simply omit the more math-heavy advanced machine learning algorithms, such as support vector machines.

4.3 Multistructural Learning Goals: Algorithm to Model

Various tasks can be involved in creating a machine learning model. This is especially true when we consider different phases of building a model, such as data preparation, training, and evaluation. Our analysis found codes that were specific to different tasks or concepts, unrelated to the model as a whole. These codes include "plan for training and testing phase by splitting data," "implement algorithm using sci-kit learn," "building baseline," and "data cleaning".

We categorized these learning goals into the multistructural stage, which is concerned with understanding several aspects of a topic independently. For example, consider the code "plan for training and testing phase by splitting data." P6 noted that test/training splitting can be a difficult concept because some students think it will be a wasteful use of data. This reflects a lack of comprehension of the bigger picture of why a test data set is essential when developing a model.

For many participants, using black box implementations aided them in teaching for a multi-structural stage. Instead of the more unistructural understanding of a simpler algorithm from the previous stage, here students need to focus on the relationship between input and output. Taking the focus away from the inner working of the algorithm also means the instructor can focus on teaching its applications. Relevant learning goals would include listing and selecting black box options, and discussing input and output.

Some participants had students implement algorithms through text-based programming with libraries (e.g., Scikit-learn) and others through visual interfaces (e.g., Microsoft Azure Machine Learning). P4 explained that libraries can be difficult for students, even if they are trained in coding, because of the need to understand the documentation and connect the available functions and their respective arguments with concepts learned in lecture. They advised that such knowledge should be taught by instructors to help their students succeed on assignments.

Participants were divided on the perceived difficulty of multistructural learning goals. For example, P2 and P5 noted that students have difficulty grasping that "sometimes all you need to know is [a function]'s arguments and what it returns" and "you don't have to develop new algorithms, but understand how to use the black box." On the other hand, P4 stated, "They can take a data set and implement things, but have difficulty in communicating and reflecting on what it means." Here we see differing perspectives from instructors on whether or not implementing algorithms with the black box is difficult.

4.4 Relational Learning Goals: Building and Tuning a Model

Our participants found it harder for students to achieve their relational learning goals. These involve a student combining their skills to build and tune a model, as well as making design decisions about features and parameters. Unlike the previous sections, there were no codes labelled as "easy" for instructors to teach. This is consistent with the observed trend that as we go up the SOLO taxonomy, learning goals become more difficult to teach.

We present three examples of learning goals at this stage:

4.4.1 Relate real life to computer input/output. In these goals, instructors discuss the creation of features. This involves combining data cleaning skills, domain knowledge of the problem, and understanding of how machine learning models are affected by the quantity and quality of features. It involves envisioning the possible features of the model. It also involves identifying how to operationalize a real world phenomenon into a format valid for computer input (e.g., going from raw text to n-grams and word vectors, going from images to numerical values).

P1 and P9 particularly noted this for natural language processing and transforming text into vectors. Per P1, "right off the bat, people struggle with turning text into vectors. They have trouble thinking about text that way." P6 commented on the difficulty of teaching this topic, "there isn't a best way to represent a variable in general— it depends on the question asked."

In the context of reinforcement learning (RL), P8 recalled working with students on a robotic arm example: "Students will gloss over the fact that the camera is input. A CS student will immediately know that the camera input will be a huge state space which you need to consider and minimize because it's too big for a computer to process in a timely manner." In general, P8 expressed RL as a difficult topic to teach, especially when students need to consider how to define state spaces, actions, and rewards.

4.4.2 Evaluate performance with consideration to the bias-variance tradeoff. The bias-variance tradeoff encompasses concepts such as overfitting/underfitting and train/test error differences. Several participants cited these elements when answering Question 4. The bias-variance tradeoff can be specifically hard to teach because the terms *bias* and *variance* have meaning outside of

machine learning. P10 explained "it's not clear what variance and bias means without a robust mathematical definition." P4 indicated that bias-variance tradeoff can be difficult because there are different types of errors to consider (e.g., train, test, approximation) which is "essentially looking at the same thing in different ways to describe and analyze a model's performance." P5 and P6 both noted that students have difficulty grasping overfitting because "sometimes training accuracy should be lower for better performance on the testing set" (P6).

4.4.3 Apply evaluation to tune parameters to improve performance. Typically, in order to optimize performance, one re-trains a model on different parameters after evaluation. Learning goals for this kind of task can involve combating effects of overfitting and underfitting, comparing and contrasting algorithm parameters, and feature selection. In our data, we found this to be a difficult task to teach, because students are surprised to learn the amount of human decision it requires.

P6 elaborated on this when asked for difficult concepts: "The idea of optimizing parameters: you have to decide when to stop optimizing. [Students] expect something more automatic with less human decision to it." P8 expressed a similar sentiment where their students believe that they do not have to do much work after implementation because the computer will figure out the rest. This may relate back to students' preconceptions of the power of machine learning from sensationalist media, believing that machines do all of the work.

Additionally, P4 noted that most students lose marks in labs when prompted to reflect on what an output means for the problem they are trying to solve. P4 cited comparing and contrasting performance on different regularization terms as an example of a difficult question for students. In terms of why this may be difficult, P5 suggested their students cannot understand how tuning works because they lack the mathematical prerequisite to understand parameters. Therefore, they claim, "they think [machine learning] is magic when you tune parameters and get different results."

4.5 Extended Abstract Learning Goals: Developing a Model to Solve a Problem

Participants described giving their students open-ended problems for assignments, class activities, and final projects, which would involve developing a new model from scratch. We categorized these open-ended problems as extended abstract learning goals because they require constructing a new model, establishing requirements for the problem, employing all the skills learned in previous stages, and then communicating the results of the model and its limitations. The two goals that we describe in further detail below were consistently found to be difficult to teach with no aspect indicated as easy.

4.5.1 Develop a model from scratch to solve a problem. The difficult aspect of this goal, as described by participants, is getting students to grasp that the design decisions of a model should be dictated by the problem they are trying to solve, rather than machine learning being "one size fits all." This could mean, as P6 expressed, that students do not evaluate their model's performance correctly in terms of the problem context (e.g., wrong evaluation metric). P10 noted that this task is difficult because the amount of algorithms can overwhelm students when deciding which to use.

4.5.2 Communicate performance and limitations of model. There are many ways to analyze a model's performance (as described in Section 4.4.2), which can make articulating it difficult. P4 stated, "[Students] can take a data set and implement things, but have difficulty in communicating and reflecting on what it means." This analysis of performance also includes describing the limits of a model and how it can be used. P9 stated, "we always have to hammer the point that [a model] will only be as good as your data." P8 noted that it is especially difficult for students to communicate their results while using concrete relevant technical terms.

Participants indicated that the issues faced at the extended abstract stage were the most difficult for their students. It was not the nuts and bolts that make ML difficult, but rather the design, evaluation, and communication of the engineering process with which their students struggle.

5 DISCUSSION

5.1 Limitations

As is typical of exploratory research, we did not structure our interview to probe for specific concepts. In our case, using the SOLO taxonomy for learning goals emerged naturally in our analysis. As our interview questions were not designed to probe specifically for these concepts, this limits the extent to which we explored and validated them.

5.1.1 Using SOLO. We began our study with the goal of identifying pedagogical content knowledge for machine learning without any consideration of learning taxonomies. Our study was exploratory; because there is so little research on machine learning education, one of our goals in the analysis was to establish or identify a theoretical/conceptual framework to work with. And in our qualitative analysis, we soon observed that the topics and learning goals that were described as easy to teach corresponded to lower SOLO/Bloom/SBF stages, and vice versa.

After making this observation, we brought the SOLO taxonomy into our analysis. The relevant codes that we reorganized into the SOLO taxonomy represented half of our data, which we found to be insightful.[2] The SOLO taxonomy was chosen over other frameworks because of its empirical basis and use in CS education research. However, because we did not enter this study with the expectation of using any particular learning taxonomy, future work is needed to explore the relationship between instructor perception of student learning and established taxonomies.

5.1.2 Interview format. We did not interview any students nor observe any classrooms, thus limiting us to the participants' perspective. Given that the present article is the first work in this area, we hope that future work will provide vital triangulation or refinement on our findings.

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. Notably, when asked Question 3, half the participants immediately responded "nothing is easy" with regards to machine learning. Most came up with examples after given some time to reflect.

5.1.3 Breadth of instructors' courses. There were considerable differences in the machine learning 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 machine learning is a new subject to be taught to non-majors, these issues arise naturally with the subject of study.

5.2 Corroborating evidence

Our work agrees with Lavesson's [21] course evaluation findings on easy and difficult topics in machine learning (described in Section 1.3). This is especially notable because Lavesson's findings present student perspectives. Furthermore, we extend his findings by presenting more depth on why certain topics are more difficult, and basing our analysis in educational psychology. We also have the advantage of sampling multiple courses within different disciplines, as opposed to one course. Lavesson's work focused on computer science students, indicating that there are similarities

[2]Most of the remaining codes represented pedagogical strategies for aiding students' transition from one SOLO level to another.

in non-major and major students' understanding of machine learning — though with regard to issues of mathematical and programming background, our data do show differences also.

5.3 Implications for Teaching

Our taxonomy, although heavily inspired by SOLO, must however be recognized as emerging from instructors' imperfect impressions of student understanding, rather than from direct inquiry of the students themselves. As such, the research described here is more likely to be useful for instructors when structuring courses rather than measuring student learning.

The SOLO taxonomy is useful for designing curricula, course materials, and assessments [8, 11]. In Section 1.2, we reviewed work that showed how learning taxonomies can be used to analyze [14, 15, 24] and improve curricula [28, 30]. By considering student development, an instructor can set and define appropriate learning outcomes for lesson plans, assignments, and exams. Given the range of our participants' courses, our classification should be sufficiently broad to allow for variation in courses for non-majors, such as length, domain-specificity, and inclusion of programming.

Because ML is a relatively new topic to be taught, with growing demand by students, there is clear motivation for research on ML education. By classifying what current instructors find easy and difficult to teach, new instructors can better anticipate what to expect when teaching machine learning non-majors and more appropriately prepare for any support they may need.

We believe the taxonomy presented is a useful framework for education researchers. For example, in our previous work on this project [29], we documented instructional techniques and strategies used by instructors for assisting students move up the SOLO taxonomy. The taxonomy was useful for us to classify pedagogical content knowledge, and helped us structure our reporting of the PCK in a way that we believe would be of interest to educators new to teaching ML.

5.4 Future Work

As a first step in widening ML education literature, we looked only at the instructor side of teaching machine learning. Future work should interview and observe students as they learn machine learning concepts of different SOLO stages. This would permit us to evaluate how such learning of machine learning develops over time. Furthermore, it would be valuable to have instructors classify their own learning goals into SOLO stages, to validate our categorization of their goals, similar to Gluga et al.'s [14, 15] work in programming courses.

More work is needed to establish what content should be covered when teaching machine learning to non-majors. We have outlined broad concepts to reach at every stage, but more work is needed on evaluating machine learning curricula. Specifically in the case of algorithms, the taxonomy presented only addresses student understanding of mathematically simpler algorithms at the unistructural stage. We did not find any patterns discussing the understanding of more mathematically complex algorithms in later SOLO stages, except implementing them using a black box approach in the multistructural stage. This could be because the majority of our participants do not cover more complex algorithms in depth, perhaps indicating a limitation of non-major student ability or that instructors are not confident in teaching these topics. However, this could also be because the duration and questions of our interviews did not provide an opportunity to probe for this. This elicits questions such as: which algorithms should be covered, and at what depth? What algorithms do instructors have trouble teaching? What pedagogical tactics can instructors use to aid them in teaching?

We noted that course content varied across our participants where the discipline of the participant is likely to be a factor. In previous work [29], the importance of teaching to the discipline of the student was expressed by the majority of our participants as a pedagogical tactic. This can look like focusing course content to relevant ML methods in the students' field, using domain-specific

examples, or giving domain-specific problems to solve. We did not analyze differences across the non-major disciplines of the participants' courses. This is worth examining as P7 made an interesting comment that student difficulty depends on disciplinary norms, citing differences in math and programming exposure across disciplines.

In our classification, we identified learning goals that instructors find difficult to teach. Future work is need to create educational tools (e.g., visualization) to support these instructional objectives. In this work, we showed that hard to teach learning goals are more related to evaluating, tuning, and communicating model performance, rather than learning goals related to specific ML algorithms. This should be considered when considering possible innovations in ML education.

6 CONCLUSION

Through interviews with ten instructors of machine learning courses for non-majors, we investigated what instructors found easy and difficult to teach. From our analysis, we noted that learning goals consistent with lower SOLO stages were reported as easier to teach, while those at the higher stage were more difficult. As the stage of SOLO increases, learning goals transition from task-specific skills to combining skills to build a model to finally applying skills to solve a problem. Participants described their students as not getting stuck on individual concepts, but rather, getting stuck on the development and evaluation of models. This is an important distinction to recognize in directing future work in machine learning education. It is also an important finding for those interested in curriculum development for non-majors. To many who are unfamiliar with AI, the algorithms are thought to be too difficult for non-majors; our findings show this is not actually a barrier. Instead, educators find it most difficult (but not impossible) to teach the design decisions that arise when applying machine learning.

There is a common perception that machine learning is inaccessible to people without proficiency in statistics and programming, and indeed this perception was reported by our participants. However, as we found through our interviews, participants describe their non-major students successfully implementing machine learning models. We hope that this work will assist in not only improving the teaching of machine learning, but in encouraging educators to teach machine learning to their non-major students.

ACKNOWLEDGMENTS

The authors would like to thank the professors and instructors who partook in interviews for their time and insights. Additionally, the authors would also like to thank the reviewers for their feedback and Horatiu Halmaghi and Jess Quynh Tran for their help coding. This work was supported by funding from the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Aziza Alsaadi. 2001. A comparison of primary mathematics curriculum in England and Qatar: The SOLO taxonomy. *Research into Learning Mathematics* 21, 3 (2001), 1.

[2] John Biggs and Catherine Tang. 2007. *Teaching for Quality Learning at University.* Open University Press.

[3] John B Biggs. 2011. *Teaching for quality learning at university: What the student does.* McGraw-Hill Education (UK).

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

[5] Benjamin S Bloom et al. 1956. Taxonomy of educational objectives. Vol. 1: Cognitive domain. *New York: McKay* (1956), 20–24.

[6] 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.

[7] Claus Brabrand and Bettina Dahl. 2007. Constructive alignment and the SOLO taxonomy: a comparative study of university competences in computer science vs. mathematics. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88.* Australian Computer Society, Inc., 3–17.

[8] 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.
[9] Francisco Enrique Vicente Castro and Kathi Fisler. 2017. Designing a multi-faceted SOLO taxonomy to track program design skills through an entire course. In *Proceedings of the 17th Koli Calling Conference on Computing Education Research*. ACM, 10–19.
[10] Helen Chick. 1998. Cognition in the formal modes: Research mathematics and the SOLO taxonomy. *Mathematics Education Research Journal* 10, 2 (1998), 4–26.
[11] Ursula Fuller, Colin G Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L Lewis, Donna McGee Thompson, Charles Riedesel, et al. 2007. Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin* 39, 4 (2007), 152–170.
[12] Yolanda Gil. 2016. Teaching Big Data Analytics Skills with Intelligent Workflow Systems.. In *AAAI*. 3997–4088.
[13] David Ginat and Eti Menashe. 2015. SOLO taxonomy for assessing novices' algorithmic design. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 452–457.
[14] Richard Gluga, Judy Kay, Raymond Lister, Sabina Kleitman, and Tim Lever. 2012. Coming to terms with Bloom: an online tutorial for teachers of programming fundamentals. In *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123*. Australian Computer Society, Inc., 147–156.
[15] Richard Gluga, Judy Kay, Raymond Lister, and Donna Teague. 2012. On the reliability of classifying programming tasks using a neo-piagetian theory of cognitive development. In *Proceedings of the ninth annual international conference on International computing education research*. ACM, 31–38.
[16] Michael Guerzhoy and Renjie Liao. 2018. Understanding How Recurrent Neural Networks Model Text. http://modelai.gettysburg.edu/2018/rnntext/index.html
[17] Mark Guzdial. 2018. Teaching Two Programming Languages in the First CS Course. https://cacm.acm.org/blogs/blog-cacm/228006-teaching-two-programming-languages-in-the-first-cs-course/fulltext
[18] Cindy E Hmelo-Silver and Merav Green Pfeffer. 2004. Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive Science* 28, 1 (2004), 127–138.
[19] Cruz Izu, Amali Weerasinghe, and Cheryl Pope. 2016. A study of code design skills in novice programmers using the SOLO taxonomy. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, 251–259.
[20] Peter Lam and Yoke-Yeen Foong. 1996. Rasch Analysis of Math SOLO Taxonomy Levels Using Hierarchical Items in Testlets. *ERIC - ED398271* (1996).
[21] Niklas Lavesson. 2010. Learning machine learning: a case study. *IEEE Transactions on Education* 53, 4 (2010), 672–676.
[22] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin* 38, 3 (2006), 118–122.
[23] Usman Mulbar, Abdul Rahman, and Ansari Ahmar. 2017. Analysis of the ability in mathematical problem-solving based on SOLO taxonomy and cognitive style. *World Transactions on Engineering and Technology Education* 15, 1 (2017).
[24] Dave Oliver, Tony Dobele, Myles Greber, and Tim Roberts. 2004. This course has a Bloom Rating of 3.9. In *Proceedings of the Sixth Australasian Conference on Computing Education-Volume 30*. Australian Computer Society, Inc., 227–231.
[25] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline L Whalley. 2008. Going SOLO to assess novice programmers. In *Acm sigcse bulletin*, Vol. 40. ACM, 209–213.
[26] Shuhaida Shuhidan, Margaret Hamilton, and Daryl D'Souza. 2009. A taxonomic study of novice programming summative assessment. In *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95*. Australian Computer Society, Inc., 147–156.
[27] 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
[28] Christopher W Starr, Bill Manaris, and RoxAnn H Stalvey. 2008. Bloom's taxonomy revisited: specifying assessable learning objectives in computer science. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 261–265.
[29] Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R Cooperstock. 2019. Can You Teach Me To Machine Learn?. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 948–954.
[30] Claudia Szabo and Katrina Falkner. 2014. Neo-piagetian theory as a guide to curriculum analysis. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 115–120.
[31] Douglas Turnbull. 2012. Music Genre Classification. http://modelai.gettysburg.edu/2012/music/
[32] Jane Watson and Jonathan Moritz. 1998. Longitudinal Development of Chance Measurement. *Mathematics Education Research Journal* 10, 2 (1998), 103–27.
[33] 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.

[34] 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.

[35] Jacqueline L Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, PK Kumar, and Christine Prasad. 2006. An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. Australian Computer Society, Inc., 243–252.