Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #47419

Developing an introductory machine learning course

Dr. Caroline Crockett, University of Virginia

Caroline Crockett is an assistant professor at the University of Virginia in the Electrical and Computer Engineering department. She received her PhD degree from the University of Michigan in electrical engineering. Her research interests include image processing and conceptual understanding.

Developing an introductory machine learning course

Abstract

This work-in-progress paper presents the design of the introduction to machine learning (ML) course at the University of Virginia. This course is targeted toward first and second-year undergraduate students and has no prerequisite courses beyond the introduction to programming course; notably, there are no linear algebra nor probability prerequisites. A key design feature of the course is that it is entry level but emphasizes the mathematical perspective of ML and conceptual understanding behind ML algorithms. This course presents the basic principles behind ML to make ML feel less like a "black box" and covers a range of applications, focusing on applications in the electrical engineering field. Students collect and interpret data, translate between textual and mathematical descriptions of systems, gain the skills necessary to implement and test ML functions in Python, and practice presenting data in easy-to-interpret plots. This paper concentrates on the set-up of the course and initial instructor reflections; we have not yet collected student data on how the course is meeting curricular goals.

1 Introduction

More engineering departments are offering, and sometimes requiring, courses on machine learning (ML). Given the breadth of the subject area, it is no surprise that these courses vary considerably. Some ML courses focus on the implementation or "how to" aspects; these courses tend to be coding-heavy and may include learning objectives familiar to many computer science courses such as learning how to interact with existing libraries. Other ML courses focus more on the mathematical theory or "how it works" aspect of ML algorithms; these courses tend to have mathematical prerequisites such as linear algebra and probability and are typically targeted for upper-level undergraduate students. Most courses blend "how to" and "how it works" learning objectives, with the amount of each depending on the level of the course and the specific learning objectives.

Many ML courses focus on third and fourth year undergraduate students, *e.g.*, [1], [2]. Some new ML curriculum design efforts, *e.g.*, [3], [4] integrated multiple ML concepts across courses, including those early in their curriculum. Regardless of the student level, most of these papers cite challenges in deciding the mix of "how to" and "how it works" learning objectives, balancing the mathematical nature of the course with its engineering applications, and/or how to reach a multidisciplinary audience.

As part of a curriculum revision at the University of Virginia, the faculty decided to introduce and require a new Introduction to ML course as part of the electrical engineering degree. The idea for the course came from faculty comments that our electrical engineering students would benefit from additional exposure to coding and that students would also benefit from a basic background in ML. The goal of this paper is to describe the educational activities and design decisions behind a course that combines both goals to share lessons learned with other instructors, especially those designing their own new ML courses.

2 Course goals

The initial goals for the intro to ML course were to increase programming practice for electrical engineering students and to expose them to ML concepts. There were two primary directions the faculty considered for the course. One option, which was ultimately rejected, was to have a "how to" focus for the course. Students would have read documentation on how to use existing ML libraries and build their own systems around these tools, considering the algorithms largely as "black boxes." This approach is enticing because it allows students to quickly see exciting applications without the need for upper-level prerequisite math courses and easily meets our goal of increasing programming practice.

The other option, which was ultimately selected, was to design more of a "how it works" ML course, where students see the details behind an ML algorithm then implement it. This option aligns better with our other electrical engineering foundational courses, such as circuits and digital logic design, which emphasize building systems from first principles. To further increase the parallels with our curriculum, the intro to ML course incorporates experimental design, with multiple assignments where students are required to collect their own data and interpret results.

The final list of learning objectives for the course are that students should be able to:

1. Thoroughly explain the workings behind simple ML classification, clustering, and regression algorithms. These should not seem like a black box!
2. Use ML terminology such as training vs testing, supervised vs unsupervised, loss functions, sparsity, hyperparameter, etc.
3. Comfortably work with arrays and matrices to represent and store data.
4. Conduct a simple machine learning experiment from writing a hypothesis, collecting data, coding a ML algorithm, to analyzing the results.
5. Present data in easy-to-interpret plots.
6. Read Python library documentation and use the information in writing new code.
7. Implement an algorithm in Python given its description in math and words.

Goals 1-2 represent the ML content in the course, goal 3 is mathematical, goal 4 is experimental, goal 5 is about technical communication, and goals 6-7 focus on coding. The coding goals emphasize the ability to work with code rather than any specific coding concepts; we do not consider the intro to ML course a valid substitute for an introductory programming course.

3 Course organization

The intro to ML course has three main parts:

1. Part 1 (6 weeks): What's inside the black box that we call ML? This part concentrates on two algorithms: nearest neighbor classification and k means. Unlike the following parts, the goal of this part is to fully understand these algorithms, including implementing them in Python.

2. Part 2 (5 weeks): How can I use ML resources? This part covers linear regression, neural networks, sparsity, and dictionary learning. The goal of this part is for students to see a variety of ML methods that they can understand most of, but they do not code these applications from scratch. Instead, students use Python libraries.

3. Part 3 (3 weeks) What other cool ML things are out there? This part briefly introduces advanced ML algorithms and the ethics of ML.

The following subsections describe the learning activities in each course part and Fig. 1 summarizes the schedule for the Fall 2024 semester. Key activities are highlighted in blue text in both the following text and in Fig. 1.

3.1 Part 1

Given the desire to present ML algorithms from first principles and the lack of prerequisite mathematics courses, we carefully selected ML algorithms that exemplify ML concepts without complicated mathematics. The k-nearest neighbor (kNN) classifier is one obvious choice; to understand this algorithm, students only need to understand vector notation and generalize a definition of distance to a multi-dimensional space. We spend roughly three weeks on background material (syllabus review, introduction to numpy, defining vectors, and coding strategies), defining the general classification problem and its applications, and the kNN algorithm.

On the first day of class, we have an introduction "class"ification activity. We ask students to stand up if their birthday is anytime January through April and define them as our training dataset. We then ask all the sitting students to find the person standing up who is closest to them, introduce themselves, ask that person what their favorite class was last semester, and keep note of whether the answer is the same as their favorite class. Next, students report out by show-of-hands if they had the same favorite class. Finally, as a class, we map out the steps involved in the activity and generalize them to the nearest neighbor algorithm. In particular, we emphasize that we load data (students enter the classroom), define a training and testing dataset (have some students stand up), loop over all test data (ask everyone not standing), find the nearest training data point (nearest standing student), and compare labels (ask for their favorite class and see if it is the same). The activity naturally leads to conversations about how to define distance, a preview for future classes.

We then introduce a 2D dataset of basketball players' heights and weights and try to classify their position. Many students have expressed that they appreciate having a consistent, simple example throughout lecture and the initial homework problems, rather than relying on abstract mathematical notation, *i.e.*, using height and weight rather than x_1 and x_2. We used a version of the freely available NBA dataset [5], where we label each player as either a forward (combines the center and forward class) or a guard, removing any players that play both positions.

After generalizing the idea of distances and kNN to multidimensional spaces via vector notation, students apply the algorithm to the handwritten digit NMIST dataset [6] in a homework problem. This requires a conversation about data representation and vectorization. For the experimental problem, students must either test the impact of (1) using different norms (2, 1, and 0-norm) for the kNN algorithm, (2) using different training dataset sizes, or (3) using the average digit of all training digits to define the training data set. In all cases, students are asked to report on the classification accuracy and make a visualization of their results.

For the majority of homework problems, we provide template code that handles all the data cleaning steps and ask students to fill in the ML algorithms and data plotting steps. We also provide links to code documentation and ask students to look up how to use library functions.

Part 1 (6 weeks): What's inside the black box that we call machine learning?		
Wed. 8/28	Syllabus, what is ML, ML applications Introduction "class"ification activity	HW 1 (code): Colab set-up
Mon. 9/2 Q01	kNN for 2d data. Distance metrics.	HW 2 (written): vectors and argmins
Wed. 9/4	kNN for n-dimensional data.	
Mon. 9/9 Q02	Intro to python.	HW 3 (code): kNN
Wed. 9/11	Coding strategies and building up to kNN.	
Mon 9/16 Q03	Introduction to cost functions and math notation for kNN.	HW 4 (code): kNN experiments
Wed 9/18	What is clustering and outline of k-means algorithm.	
Mon 9/23 Q04	K means cost function.	HW 5 (code): k means
Wed 9/25	Color compression for images	
Mon 9/30 Q05	Review homeworks 1-5.	Exam corrections
Wed. 10/2	**Exam 1**	
Part 2 (5 weeks): Using ML libraries		
Mon 10/7 Q06	Define regression. Linear regression with one predictor.	HW 6 (written): linear regression
Wed. 10/9	Expanding to two variables. Matrix multiplication.	
10/14-15: Fall reading days		
Wed 10/16	Lin reg: generalizations. Review matrix-vector multiplication.	HW 7 (code): linear regression
Mon 10/21 Q07	Perceptron. Neural networks (NN) intro.	HW 8 (code): Neural networks
Wed. 10/23	Neural network terminology	
Mon 10/28 Q08	Sparsity and denoising.	HW 9 (written): sparsity, dictionaries
Wed. 10/30	Sparse representation for music.	
Mon 11/4 Q09	Music denoising	Project proposal
Wed. 11/6	Learning a dictionary.	
Mon 11/11 Q10	Review homeworks 6-10.	Exam corrections
Wed. 11/13	**Exam 2**	
Part 3 (3 weeks): What other cool machine learning things are out there?		
Mon 11/18	Hyperparameter optimization	HW 11 (written): HPO
Wed. 11/20	Ethics in ML: applications	Project overview and slides due. Project reviews and reflection due.
Mon 11/25 Q11	Ethics in ML: research considerations	
	Thanksgiving break	
Mon 12/2 Q12	Project work time	
Wed. 12/4	Project work time	
	Final exam timeslot: Project presentations	

Figure 1: Example course schedule. Q## represent weekly review quizzes. These are low-stakes opportunities for constructive feedback and automatically graded using a learning management system. Anecdotal evidence suggests students find the hands-on activities particularly motivating and fun.

The next algorithm in the course is the k-means clustering algorithm, which naturally builds on the distance and vector notation from kNN. K-means involves iterating between a centroid update step (calling a mean function) and a class update step (a call to the kNN algorithm). This progression lends itself to a conversation about the differences between system and sub-system design princi-

ples and how it is important to define the input-output specification for each block of a system; these principles are shared with many other intro-level electrical engineering course.

In-class, we discuss the RGB representation of color and how we can include that feature information in a vectorized image. Starting in-class and finishing as a homework problem, students apply k-means to the problem of color compression for images. Fig. 2 shows an example result.

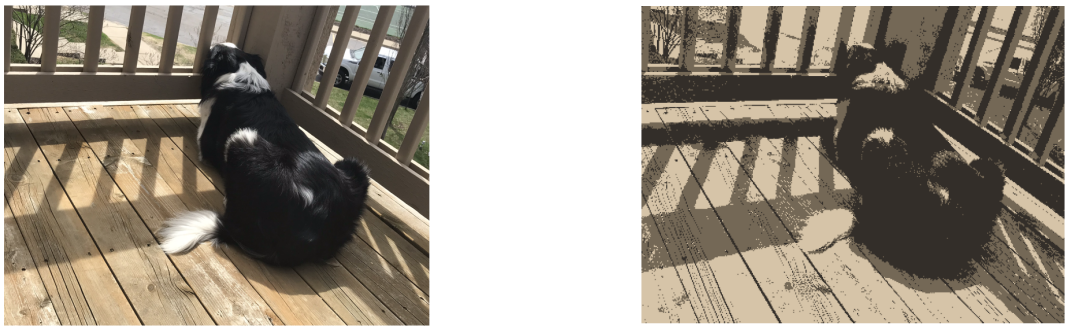

Figure 2: Original image and its color compressed version with $k = 3$ to demonstrate an application of the k-means algorithm.

3.2 Part 2

Part 2 of the course covers linear regression, the basics of neural networks, and dictionaries. The homeworks emphasize reading documentation and using existing libraries rather than coding algorithms from scratch.

Many students have seen linear regression before in a variety of settings and they typically understand the mathematics behind simple linear regression reasonably well. It is easy to select an electrical engineering application for homework problems, *e.g.*, modeling the voltage-current relationship of a MOSFET in a given region as linear. The experimental problem on the linear regression homework looks at the impact of adding either a single outlier at varying distances or adding a constant outlier but varying the number of other datapoints.

Given its familiarity, linear regression serves as a good starting point for introducing the perceptron. We frame the perceptron as a linear regression operation followed by a non-linear function. From here, we can define the multilayer perceptron.

There are many ML topics related to neural networks that we do not have time to fully cover in the class but would like to expose students to. To do so efficiently, we have a neural network terminology activity where students are split into groups of about 3 and we give each group 10 minutes to look up and write a definition for an assigned concept on a chalkboard (or a paper taped to the wall) spread around the room. Example concepts are: backpropagation, stochastic gradient descent, momentum - in the context of gradient descent, early stopping for gradient descent, batches, max pooling layer, tensor, active learning, autoencoder, fully connected layer, and validation data. The groups then rotate and draw a picture for their new concept. After one more rotation, students prepare to explain their new concept to the class, using only the information from the previous groups. Once all groups present (1 minute each), the class discusses what made some definitions and visualizations easier to understand and takeaways for their final project presentations.

Perhaps the most surprising topics we cover in the intro ML are sparsity and dictionaries. Dictionary learning is mathematically advanced and even the general denoising problem is non-trivial for an arbitrary dictionary. However, the underlying concepts are attainable for early undergraduate students and the math is simple if you limit the discussion to an invertible dictionary. Conveniently, electrical engineering has a very useful invertible dictionary: the Fourier transform. Although using the Fourier transform to denoise a signal is not ML, the concepts of data representation and sparsity are key to many more advanced ML algorithms.

To build up to denoising a signal using the Fourier transform, the class defines the zero "norm" and thresholding operations, then demonstrates denoising a sparse edge image. While most signals are not sparse, they can often be represented in a domain where they are sparse, such as the frequency domain. Many of our students are interested in music, and the idea of writing a chord as a sum of frequencies is intuitive for them even if they have not taken a signals and systems course. Fig. 3 shows the motivating example for the sparse audio signal denoising.

For the dictionary music denoising experiment, the students gather their own audio file using tuning forks to approximate pure tones and whatever noise they decide to add, then denoise it using provided code functions for transforming between the time and frequency domains (the course does not discuss phases, complex numbers, or negative frequencies and the provided code abstracts these details away). Anecdotally, students are typically very surprised at how well the method works when comparing their noisy and denoised audio signals.

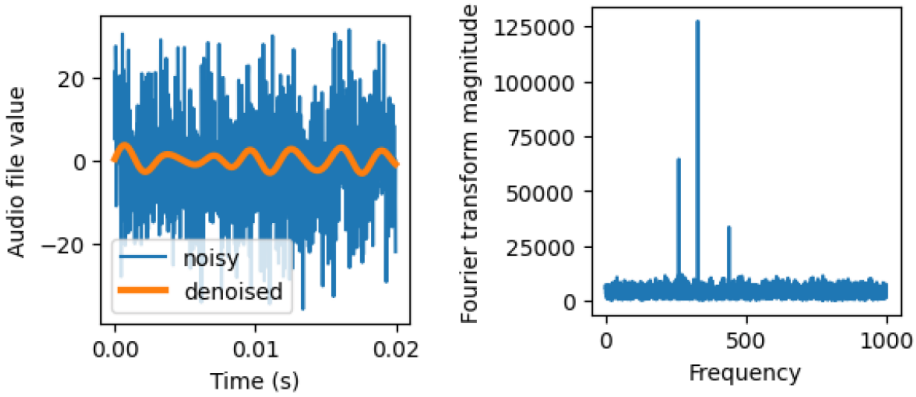

Figure 3: Example of denoising an audio signal by sparsifying its frequency representation using the Fourier transform. The clean signal is the sum of three pure tones.

3.3 Part 3

In the final few weeks of the semester, the students primarily work on a group project. In-class time is split between lectures introducing advanced topics and group work time. The lectures are high-level only and do not present the full "how it works" behind these algorithms.

Due to student interest in the topic, the course includes one day on hyperparameter optimization. We define the common brute force, grid search, and random search methods. We also discuss Figure 1 from [7] showing that random sampling explores more distinct values than grid search and thus often discovers a better hyperparameter. Students are often surprised (and even disappointed)

at the simplicity of these common methods. When there are many hyperparameters or when the cost of trying a hyperparameter is expensive, a designer might instead opt for a more complex strategy such as Bayesian optimization [8] or gradient descent for hyperparameter optimization [9]. We briefly present these two strategies at a high level, using pictures of a 1D cost function for illustration. Although the details of both are outside the scope of an introductory course, students quickly relate to the exploitation-exploration trade-off inherent in Bayesian optimization when presented with an example such as them being more likely to try new restaurants as a first-year and go to a favorite restaurant in their last semester.

The course includes two class periods on ethics in ML. In the first class, to highlight student perspectives, students form groups, research a topic, and present to the class (this is all done in a single class). Suggested topics are the energy use of ML, training data for speech recognition, auto grading of student exams, predicting repeat criminal behavior [10], and phantom braking in self-driving cars [11], but students are also invited to pick their own topic. The second class is based around ethical considerations in ML research and covers publication bias, hypothesizing after the results are known [12], [13], biased training data, the reproducibility crisis [14], "grad student" descent [12], confusing explanation and speculation in papers [15], excess "mathiness" [15], and inaccurate language [15]. Students hypothesize about the root causes of these ethical issues and possible solutions then compare their answers to those presented in [16]. This lecture has been well received, with many students citing the ethics week as one of their favorite topics in the course.

4 Conclusion

More engineering departments are offering a variety of ML courses to meet the interests of students, in response to the needs of industry, and to prepare future engineers to design and use ML tools. ML is inherently interdisciplinary, requiring programming and computer science background, mathematical knowledge, and concepts from signal and image processing, traditionally taught in electrical engineering. Courses in ML are thus unsurprisingly varied in how they approach the topic. For example, some courses focus more on using ML as a tool and expect students to be able to read documentation from common ML libraries such as pytorch. Other courses require a strong mathematical background and focus on the theoretical underpinnings of algorithms such as gradient descent and linear regression.

The introduction to ML course presented in this paper was designed to fit in an electrical engineering curriculum where we want students to understand some ML concepts from first principles, but without requiring upper-level mathematical prerequisites. The only expectation of students coming into the course is an introductory course in Python. We have carefully chosen the course content to expose students to exciting areas of ML and a variety of applications.

One avenue for future work is assessing the effectiveness of the intro to ML course at meeting its goals. In an informal survey at the end of the first course offering, most students agreed that the course improved their understanding of math expressions, their coding ability, and their understanding of ML concepts. In addition to collecting further student survey data on their perceptions of learning, a tool such as the Engineering Computational Thinking Diagnostic (ECTD) [17] would be useful to more rigorously evaluate whether students' improved their computational

thinking skills.

Another avenue of future work is comparing the intro to ML course to ML courses at other universities by looking at types of assignments, opportunities for project-based learning, course prerequisites, mode of instruction, major department, class size, typical year of enrolled students, and whether the course is required for any degree program, and the learning platforms utilized. This could take the form of a systematic review, inspired by the work of Marques *et al.* [18] at the K-12 level on systematically reviewing 30 ML instructional units. Given the quickly changing landscape of ML courses, any survey of ML courses requires contacting instructors rather than relying on information from departmental websites. We expect a wide range of approaches, each with their own advantages and disadvantages. Seeing the different direction of ML courses can aid instructors and curriculum developers select the correct learning goals and activities for their specific student population.

References

[1] S. Isaac Flores-Alonso, N. V. M. Diaz, J. Kapphahn, *et al.*, "Introduction to AI in undergraduate engineering education," in *2023 IEEE Frontiers in Education Conference (FIE)*, College Station, TX, USA: IEEE, Oct. 18, 2023, pp. 1–4, ISBN: 9798350336429. DOI: 10.1109/FIE58773.2023.10343187.

[2] S. Khorbotly, "Machine learning: An undergraduate engineering course," in *2022 ASEE Illinois-Indiana Section Conference Proceedings*, Anderson, Indiana: ASEE Conferences, Apr. 2022, p. 42 132. DOI: 10.18260/1-2--42132.

[3] R. DeMara, A. Gonzalez, A. Wu, *et al.*, "A crcd experience: Integrating machine learning concepts into introductory engineering and science programming courses," in *2003 Annual Conference Proceedings*, Nashville, Tennessee: ASEE Conferences, Jun. 2003, pp. 8.36.1–8.36.23. DOI: 10.18260/1-2--12117.

[4] L. Huang, "Integrating machine learning to undergraduate engineering curricula through project-based learning," in *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA: IEEE, Oct. 2019, pp. 1–4, ISBN: 978-1-72811-746-1. DOI: 10.1109/FIE43999.2019.9028688.

[5] stats.nba.com, *NBA Database*. [Online]. Available: https://www.kaggle.com/datasets/wyattowalsh/basketball.

[6] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov. 2012, ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2211477.

[7] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, Feb. 2012, Number of pages: 25 Publisher: JMLR.org tex.acmid: 2188395 tex.issue_date: 3/1/2012, ISSN: 1532-4435. DOI: 10.5555/2188385.2188395.

[8] P. I. Frazier, "A tutorial on bayesian optimization," Jul. 2018. [Online]. Available: https://arxiv.org/abs/1807.02811.

[9] C. Crockett and J. A. Fessler, "Bilevel methods for image reconstruction," *Foundations and Trends® in Signal Processing*, vol. 15, no. 2, pp. 121–289, 2022, ISSN: 1932-8346, 1932-8354. DOI: 10.1561/2000000111.

[10] E. Yong, *A Popular Algorithm Is No Better at Predicting Crimes Than Random People*, en, Section: Technology, Jan. 2018. [Online]. Available: https://www.theatlantic.com/technology/archive/2018/01/equivant-compas-algorithm/550646/.

[11] B. Templeton, *An 8-Car Pileup Started By A Tesla In Autopilot Opens Up Many Complex Issues*, Jan. 2023. [Online]. Available: https://www.forbes.com/sites/bradtempleton/2023/01/11/an-8-car-pileup-started-by-a-tesla-in-autopilot-opens-up-many-complex-issues/?sh=2fae2f0212be.

[12] O. Gencoglu, M. van Gils, E. Guldogan, *et al.*, "HARK side of deep learning – From grad student descent to automated machine learning," en, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1904.07633.

[13] N. L. Kerr, "HARKing: Hypothesizing After the Results are Known," *Personality and Social Psychology Review*, vol. 2, no. 3, pp. 196–217, Aug. 1998, Publisher: SAGE Publications Inc, ISSN: 1088-8683. DOI: 10.1207/s15327957pspr0203_4.

[14] S. Kapoor and A. Narayanan, *Leakage and the Reproducibility Crisis in ML-based Science*, Jul. 2022. [Online]. Available: http://arxiv.org/abs/2207.07048.

[15] Z. C. Lipton and J. Steinhardt, "Troubling Trends in Machine Learning Scholarship," en, *Queue*, vol. 17, no. 1, pp. 45–77, Feb. 2019, ISSN: 1542-7730, 1542-7749. DOI: 10.1145/3317287.3328534.

[16] M. R. Munafò, B. A. Nosek, D. V. M. Bishop, *et al.*, "A manifesto for reproducible science," en, *Nature Human Behaviour*, vol. 1, no. 1, p. 0021, Jan. 2017, ISSN: 2397-3374. DOI: 10.1038/s41562-016-0021.

[17] N. V. Mendoza Diaz, S. Y. Yoon, D. A. Trytten, and R. Meier, "Development and validation of the engineering computational thinking diagnostic for undergraduate students," *IEEE Access*, vol. 11, pp. 133 099–133 114, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3335931.

[18] L. S. Marques, C. Gresse Von Wangenheim, and J. C. R. Hauck, "Teaching machine learning in school: A systematic mapping of the state of the art," *Informatics in Education*, pp. 283–321, Jun. 15, 2020, ISSN: 1648-5831, 2335-8971. DOI: 10.15388/infedu.2020.14.