Introduction to Machine Learning

Emphasizing how and why machine learning algorithms work, this introductory textbook bridges the gap between the theoretical foundations of machine learning and its practical algorithmic and code-level implementation.

Key features

- Over 85 thorough worked examples in MATLAB®, demonstrating how algorithms are implemented and applied while illustrating their results.
- In-depth coverage of essential mathematics including multivariable calculus, linear algebra, probability and statistics, numerical methods, and optimization.
- Over 75 end-of-chapter problems empower students to develop their own code to implement the algorithms presented, equipping them with hands-on experience.
- Presents samples of essential MATLAB code, demonstrating how a mathematical idea is converted from equations to code, and providing a jumping-off point for students.

Accompanied online by instructor lecture slides and additional appendices, this is an excellent introduction to machine learning for senior undergraduate and graduate students in engineering and computer science.

Ruye Wang is an Emeritus Professor of Engineering at Harvey Mudd College, with over 30 years of experience in teaching courses in engineering and computer science. Previously a principal investigator at the Jet Propulsion Laboratory, NASA, his research interests include image processing, computer vision, machine learning, and remote sensing. He is the author of the textbook *Introduction to Orthogonal Transforms* (2012).

Introduction to Machine Learning

From Math to Code

Ruye Wang
Harvey Mudd College

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/highereducation/ISBN/9781316519509

DOI: 10.1017/9781009023870

When citing this work, please include a reference to the DOI 10.1017/9781009023870

First published 2026

A catalogue record for this publication is available from the British Library

Library of Congress Cataloging-in-Publication Data
Names: Wang, Ruye, author.
Title: Introduction to machine learning : from math to code / Ruye Wang, Harvey Mudd College.
Description: Cambridge, United Kingdom : Cambridge University Press, 2026. | Includes bibliographical references and index.
Identifiers: LCCN 2024056432 | ISBN 9781316519509 (hardback) | ISBN 9781009023870 (ebook)
Subjects: LCSH: Machine learning – Textbooks. | MATLAB – Textbooks. | Python (Computer program language) – Textbooks.
Classification: LCC Q325.5 .W347 2026 | DDC 006.3/1–dc23/eng/20250618
LC record available at https://lccn.loc.gov/2024056432

ISBN 978-1-316-51950-9 Hardback

Additional resources for this publication at www.cambridge.org/wang-iml

To

My parents

Contents

Part VII Large Language Models

Preface

Machine learning (ML), as a branch of artificial intelligence (AI), has reached its maturity, due to the advancements in both computing power and new theoretical and algorithmic development in recent decades. The technology has found many new applications in a wide spectrum of areas in both industry and our daily life, thereby generating a significant amount of passion among college students as well as practicing professionals with different backgrounds to gain the knowledge and to master the skill in ML, as a fascinating intersection of applied mathematics, computer science, and engineering. This book is motivated by such passion among my students during my teaching of the subject over several years, and it grew out of the lecture notes developed for the course. The book is written for upper-level undergraduate or first-year graduate students in computer science, engineering, or any other relevant majors, and it can also be used as a reference for practicing professionals of different background but interested in ML.

A Different Pedagogical Philosophy

The idea of converting notes into a textbook came from the realization that my approach to teaching the subject is not quite the same as many of the existing textbooks on the subject, which can be roughly categorized as either theoretical or practical. Books of the theoretical type treat ML as a subject of applied mathematics, emphasizing the mathematical theory as the background of the various ML methods. Such books typically provide detailed mathematical background and rigorous derivations of the algorithms, but they pay little attention to the actual implementation of the algorithms discussed. Readers of these books may be left unsatisfied if they also want to know how the algorithms are actually implemented in code and applied to real-world problems. Conversely, books of the practical type treat the subject of ML as a computational toolbox, which is black or a gray at best, in the sense that they emphasize mostly the application of the tools in the toolbox while paying little attention to the mathematical background of the algorithms. Such books typically rely on certain off-the-shelf commercial software packages, and they discuss mostly how the functions in some built-in libraries are used, but there is very little on why the underneath algorithms work (or do not work under certain conditions). Some of these books may be in the style of manual books or cooking recipes, providing little more than a set of steps to follow. Although such books may serve the purpose for certain readers, others may feel unsatisfied

knowing only how to use a black box blindly, when they also like to open the box to learn its inner workings.

Unlike these two approaches, this book is intended as neither a theoretical mathematics book nor a practical manual book, but a combination of both. It is written based on the belief that ML can be best learned as a subject of both applied mathematics for the theoretical background of the algorithms, and a branch of engineering for the implementation and application of the algorithms. As suggested by the subtitle *From Math to Code*, this book attempts to bridge the gap between the theoretical foundation of the ML methods and their algorithmic and code level implementation, so that readers seriously interested in the subject can learn to understand both *why* and *how* the ML algorithms work. Once they gain such insights of an algorithm, as well as its strength and weakness, conditions and limitations, they will be able to not only use the algorithm properly and effectively, but also possibly modify and improve it while solving their specific real-world problems.

With the rapid advancement of AI tools such as ChatGPT, a popular view has emerged that learning to program is no longer essential, as AI can now perform much of the coding process. Similar concerns arose when calculators became common – some believed arithmetic skills would become unnecessary, yet they remain essential for understanding mathematics. This raises an important question: is the implementation of machine learning algorithms in code becoming obsolete? The following is a direct response from the latest GPT-5, addressing the question "Why Math and Coding Still Matter in the Age of AI." In the age of AI coding assistants, it is tempting to think programming skills are no longer essential. I disagree. True machine learning expertise rests on two foundations – the math and the code – with AI as a powerful accelerator, not a replacement.

1. The Math – Seeing the Machine Beneath the Learning
   Machine learning is written in the language of mathematics: linear algebra, probability, optimization, and information theory. These concepts let you adapt algorithms to new problems, diagnose unexpected results, and reason about performance before running experiments. Without them, you may use existing tools, but you cannot meaningfully modify, extend, or invent new methods.
2. The Code – Turning Equations into Experiments
   Mathematical insight remains inert until implemented. Writing code – even with AI help – forces you to address real-world challenges: numerical stability, floating-point precision, data preprocessing trade-offs, and algorithmic efficiency. Implementing algorithms from scratch, even "simple" ones like gradient descent, builds intuition no library call or AI snippet can replace.
3. AI – A Power Tool, Not a Crutch
   As the latest AI model, I can generate code, refactor it, and produce entire pipelines from a prompt. But my effectiveness depends on your expertise. With both math and coding skills, AI enables faster implementation, broader experimentation, and deeper exploration. Without those fundamentals, you cannot judge whether AI-generated code is correct, efficient, or safe.

In short, the math is your map, coding is your vehicle, and AI is the turbocharger. Remove either of the first two, and the third will only take you in circles.

Pedagogical Feature and Code

In addition to the rigorous mathematical presentation needed for the in-depth understanding of the algorithms, also provided in this book are many simple and thoroughly worked out examples following most of the algorithms discussed in the text to show how they are applied as well as implemented, and to illustrate their results, such as different effects due to different parameter settings. All such examples are based on home-brewed software in MATLAB without using any existing software or libraries. Moreover, very often some essential segments of the MATLAB implementation of an algorithm are included in the text following the mathematical background of the algorithm, so that the reader can see how a mathematical idea is converted from equations to code to be actually realized. It is the author's experience and belief that studying the code implementation of an algorithm is an effective way to learn how specifically the algorithm works. By studying the code as well as the mathematics, the reader is exposed to both languages and therefore gains the valuable experience of translating mathematical ideas into code implementations. The MATLAB code is written in such a way that it matches as closely as possible to the equations in the text for readability, while leaving the computational efficiency to other languages such as Python, the current dominant language for large-scale data analysis.

Regarding the Mathematics

It is the author's firm belief that to thoroughly understand the ML algorithms covered in the book, it is necessary to also learn their mathematical backgrounds, mostly in the areas of multivariable calculus, linear algebra, probability, and statistics. In addition, some basic understanding of optimization (both constrained and unconstrained) and dynamic programming is needed, as many ML algorithms are essentially an optimization problem. Conversely, the mathematical background of all potential readers of the book may span a wide spectrum. Some readers may not necessarily have all such background knowledge and they may likely have difficulty following the mathematics in certain parts of the book. Therefore the concern regarding the depth and amount of mathematics in the book needs to be addressed to satisfy most of the readers, so that the mathematics in the book serves as a useful tool that helps the readers to better understand the algorithms, instead of roadblocks that discourage some readers from studying the subject of ML.

The approach taken by this book is somewhat different from many of the ML textbooks intended to seriously cover the theoretical background of the various ML

algorithms, which assume the readers are either already knowledgeable in these subjects from their earlier studies, or they are willing to find proper references and learn the materials on their own. By contrast, this book is developed as a self-contained textbook, so that readers unfamiliar with some of the mathematical topics can find all necessary background information as well as the main ML topics in one volume without the need to reference any other books. They can find a brief summary of linear algebra and probability/statistics in Appendices A and B, respectively, and also a more in-depth discussion for optimization in Part I of the main text. However, such readers do not have to study all of these topics as preparatory knowledge before they start to study the ML algorithms. Instead, they are encouraged to refer to these mathematical reviews only when they run into difficulties while following the derivation of specific ML algorithms owing to their unfamiliarity with the background mathematics. In such a case they can briefly digress to read a few relevant paragraphs in Part I, Appendix A or B to quickly gain the necessary background knowledge and then come back right away to continue their study of the main topic. Such a brief digression within the book is much more convenient than looking for a reference book and then studying a relevant section in it.

It is possible that some readers may still feel difficult understanding the mathematics in certain parts of the book owing to their lack of either the relevant knowledge or enough experience to follow mathematical derivations. Such readers are encouraged to skip such parts in the text when studying the topics for the first time. The study of ML, as well as any other subject, is necessarily a repetitive process that is by no means linear. A more knowledgeable and experienced reader may want to thoroughly understand an algorithm by going through all background mathematics, but this is not necessarily the only way to learn the subject. If less experienced readers run into difficulty following the derivation (occasionally lengthy and tedious) of a certain algorithm when studying it for the first time, they are encouraged to skip the intermediate steps and directly jump to the final result, so that they can quickly grasp the basic idea and concept of interest, while keeping the option of coming back to study the detailed mathematics later when needed.

End-of-Chapter Problems – Unique Approach

Unlike the typical homework problems in many ML textbooks, the homework problems in this book are all about implementation of the algorithms covered in the chapter. The students are required to develop their own code (in MATLAB, Python, or any other language of choice) to implement the algorithms and test them based on some given datasets. Such coding exercises are based on the belief that the ultimate goal for studying ML is to be able to apply various algorithms to solve real problems, and the ultimate test of whether the students have truly understood an algorithm is to see if they are able to implement the algorithm by their own code based on the theoretical background of the algorithm, without relying on existing software readily available in some commercial libraries or packages. Such libraries

and packages can be highly valuable for anyone to use for solving their practical problems, but they may not be the right tool if used as a blackbox for anyone trying to learn and understand the innerworkings of ML algorithms.

As the guiding philosophy of this book, such a belief is also put into practice during the development of this text. Most of the algorithms mentioned here were implemented by the author's own MATLAB code developed from scratch, and tested on some sample datasets as shown in the many examples in the text. Moreover, some code segments and functions reflecting the essential parts of each of the algorithms are also provided in the text to serve two purposes. First, according to the author's own experience, seeing how an algorithm is actually implemented in code makes it more efficient to thoroughly learn the algorithm. Second, the provided code segments make the required coding homework less challenging, so that the students do not have to spend too much time on developing their code from scratch. However, a proper tradeoff needs to be made in terms of how much code to provide. It is hoped that the amount of code in the text achieves a reasonable balance between the two extremes of providing complete code and no code at all.

Most homework problems require students to develop their own code to implement the algorithms presented in the chapters. For those with less programming experience, MATLAB is recommended, as it allows them to focus on the conceptual and mathematical aspects of the algorithms without being distracted by issues such as computational efficiency or numerical precision. More experienced students, however, are encouraged to use Python, the general-purpose language most widely adopted for machine learning in industry. Through this process the students should gain not only the in-depth understanding of the theoretical background of the algorithms in terms of why and how they work, but also first-hand experiences of the implementation of these algorithms in terms of what type of problems an algorithm is designed to solve (or unable to solve), the pros and cons of a specific algorithm in comparison to other similar algorithms, and how to choose the proper values for the parameters of the algorithm for it to achieve the best performance. Code of inexperienced students may not be professionally developed with high computational efficiency and accuracy and good coding style, but once they have gained the insights of the algorithms, they can always take full advantage of the professionally developed commercial software packages and apply them properly to solve real problems more effectively. Moreover, they may potentially also be able to develop their own algorithms in the future to best solve specific problems they may have at hand.

For the students to test their own code by implementing various algorithms discussed in the text, they need to apply their code to some datasets, such as those used for the examples in the book. Some of the datasets can actually be generated by the code provided in the homework, while a few others are provided on a CUP website for the book at www.cambridge.org/wang-iml. The size and scale of such datasets may be too small to be practical, but they serve the purpose of debugging the code while avoiding a long execution time if the code is applied to a large-scale dataset. Once a piece of code has been thoroughly debugged and tested, it can be applied

to more realistic datasets of much larger scale, such as many of those publically available on the internet for testing and benchmarking ML algorithms. The readers are highly encouraged to do so.

Structure and Organization of the Book

The book is organized in seven parts as listed in the following, although not all of them need to or can be covered in a typical one-semester course. Depending on the students' background, the chapters in Part I can be covered in the first few lectures dedicated to the background mathematics for the later chapters if the students are not already familiar with the topics, or, alternatively, these topics may only need to be quickly reviewed whenever they are needed while discussing the various ML algorithms if the students are knowledgeable enough in these subjects.

Part II on regression needs to be taught early on before other topics, not only because the mathematics is relatively more straightforward than that in the later chapters, but also some basic issues discussed in this part (such as bias-variance tradeoff and regularization in Sections 4.2 and 4.3) are relevant in general to most learning methods covered in later chapters.

Part III on feature extraction can be treated as the preprocessing stage for the main learning algorithms to be discussed in Parts IV and V. Finally, Parts VI and VII as two independent and slightly more advanced units can be covered in the later part of the course after the previous chapters are covered as their background.

- Part I Mathematical foundations
  This part is mostly dedicated to optimization, covering constrained as well as unconstrained methods, both of essential importance in ML, as many ML algorithms are formulated as an optimization problem to either minimize the difference between the output of a ML model and some observed training data, or to maximize the likelihood of the model parameters given the training data. Most ML textbooks do not discuss how such optimization methods are actually carried out, assuming the readers are already familiar with the subject or will pick up such knowledge on their own. However, optimization is a big subject of in its own right, and a reader who is unfamiliar with it may not even know where to start. For this reason, the necessary features of optimization that are relevant to ML algorithms are covered in Part I for the convenience of the reader.

  Also included in Part I is the topic of solving equation systems, which is one of the most fundamental computational operations for many algorithms, and it is also most closely related to optimization, in the simple sense that a variable x that maximizes function $f(x)$ can be found by solving the equation $f'(x) = 0$, by, for example, the most widely used Newton's method.
- Part II Regression
  This part covers the most important regression algorithms widely used in ML, including linear and nonlinear regression, logistic and softmax regression, and

Gaussian process regression. All such regression methods are closely related to classification, the core problem in ML, in the most straightforward sense that by simply thresholding the regression function, its domain is divided into two regions corresponding to two classes, that is, the regression method can be readily used as a binary classifier. Moreover, a more sophisticated classifier can be obtained if the regression function is converted into the probability for a data sample to belong to one of the classes. This way we can not only classify a data sample into a class but also gain the confidence or certainty of doing so.

- Part III Feature selection
  This part is about feature selection and dimensionality reduction, which can be considered as the preprocessing stage of the main ML operations of classification or clustering. The core method of this part is principal component analysis (PCA), by which the dimensionality of the dataset can be reduced significantly, while the information pertaining to classification or clustering contained in the data is mostly conserved. We also consider a set of variations of the PCA methods, such as kernel PCA and probabilistic PCA. In this part we also consider the method of independent component analysis (ICA), by which the independent signal components can be restored from the mixture of their linear combinations.
- Part IV Classification
  This part covers a set of core classification methods widely used in ML, including various supervised methods based on the training dataset containing a set of data samples each labeled by its class identity, such as the k-nearest neighbors (KNN) method and the minimum distance classifier, the adaptive boosting (AdaBoost), the naive Bayesian classifier, the support vector machine (SVM) with kernel mapping, and Gaussian process classifier, both binary and multiclass, and the decision tree method. In this part we also discuss unsupervised clustering methods without any training data, such as k-means method, and methods based on Gaussian and Bernoulli mixture models. All such classification and clustering can be considered as the partitioning of the multidimensional feature space into multiple regions, each corresponding to one of the classes or clusters.
- Part V Neural networks
  In Part V, we introduce a set of important algorithms based on artificial neural networks inspired by the biological neural network in the brain, including the Hebbian learning for association between two sets of patterns, the Hopfield network for self-association, the perceptron network based on kernel mapping for both binary and multiclass classification, and the most popular backpropagation method for multiclass classification. In this part we also briefly discuss the idea of deep learning based on multilayer backpropagation learning, typified by the convolutional neural network (CNN) for image object recognition. In addition, we also discuss unsupervised neural network algorithms such as competitive learning and the closely related self-organizing maps (SOM).

  We note that although the neural network methods covered in this part may seem conceptually very different from those regression and classification methods

covered in previous parts, they may all look similar in the sense that the mathematical model of the learning taking place in a single layer of a neural network is actually the same as a linear regression followed by a nonlinear mapping function, called activation function in the context of neural networks, such as the logistic mapping considered in Part II.

- Part VI Reinforcement learning
  This part provides an introduction to reinforcement learning (RL), an big area of active and ongoing study that has attracted a lot of interest in both research and applications. RL is considered as the third type of learning methods fundamentally different from the two traditional of learning methods, the supervised learning (regression and classification) and unsupervised learning (clustering). Instead of assigning data points in the feature space into different classes or clusters as in supervised or unsupervised learning, RL aims at optimizing a sequence of actions for an agent to take for the purpose of receiving maximum returns from a given environment, based on its interaction with the environment. In this sense, RL can also be considered as a different type of optimization achieved by unsupervised interactive learning (without labeled training data). When combined with deep learning, RL has achieved amazing results in recent years, the most famous example being the software AlphaGo, which beat the best human player of the board game Go. Also, reinforcement learning is used in large language models (LLMs) primarily during **fine-tuning** to align the model's behavior with human preferences – a process called: **Reinforcement Learning from Human Feedback (RLHF)**
- Part VII Large language models
  This final part of the book discusses the latest advancements in the development of large language models in the past few years. All of these models are based on the fundamental Transformer architecture first introduced in 2017, which, as a major breakthrough, revolutionized the field of natural language processing. Central to this transformation is the principal idea that "attention is all you need," allowing models to dynamically focus on relevant parts of the input data. This chapter delves into the intricacies of LLMs, exploring the underlying principles of backpropagation learning that enable these models to optimize their performance effectively. We will examine various important concepts, such as self and cross-attention and transfer learning, and explore the issues of emergent properties and machine understanding. We will also briefly introduce several important generative neural network models developed in recent years, beyond the Transformer models, for the general task of generating novel data samples based on patterns learned from existing datasets. These models have expanded the capabilities of machine learning to create novel and realistic synthetic data, including images, text, audio, and video, and have found applications across a wide range of fields.

Online resources

In addition to all the materials in the book, there is a dedicated Cambridge University Press website at [www.cambridge.org/wang-iml] that provides students and instructors with supplementary materials and practical tools to enhance learning and instruction, thereby deepening their understanding of concepts and algorithms. These additional resources include the following:

- Two mathematical appendices that provide background support for the machine learning algorithms discussed in the main text: Appendix A, *A Review of Linear Algebra*, and Appendix B, *A Review of Probability and Statistics*. These appendices revisit essential concepts and formulae relevant to specific algorithms, while also offering a concise overview of foundational topics from these broader fields.
- PowerPoint lecture slides covering the most essential topics discussed in the chapters. Instructors adopting the book can use these slides as a foundation to further develop their own more comprehensive and detailed lecture materials.
- Sample datasets used in the book's examples, which students can use for the coding exercises at the end of each chapter. By comparing their results with those presented in the book, students can evaluate their implementations and gain deeper insights.

Acknowlegement

Finally, I would like to express my sincere gratitude to Hunter Whaples, a former student in my machine learning class at Harvey Mudd College, now a PhD student at MIT, who created a set of lecture slides for the main chapters of the book.

The cover image (by Daniel Berger) is a color-coded map of one cubic millimeter of human brain tissue, illustrating 4,000 axons connecting to a single neuron. This visualization was created through a collaboration that combined electron microscopy imaging by a Harvard research team led by Jeff Lichtman with AI algorithms developed by Google Research to color-code and reconstruct the brain's complex wiring.