O'REILLY®

Fundamentals of Deep Learning

DESIGNING NEXT-GENERATION MACHINE INTELLIGENCE ALGORITHMS

Nikhil Buduma
with contributions by Nicholas Locascio

Fundamentals of Deep Learning

Designing Next-Generation Machine Intelligence Algorithms

Nikhil Buduma

with contributions by Nicholas Locascio

Beijing · Boston · Farnham · Sebastopol · Tokyo

O'REILLY®

Fundamentals of Deep Learning
by Nikhil Buduma and Nicholas Lacascio

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Mike Loukides and Shannon Cutt
Production Editor: Shiny Kalapurakkel
Copyeditor: Sonia Saruba
Proofreader: Amanda Kersey
Indexer: Wendy Catalano
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

June 2017: First Edition

Revision History for the First Edition
2017-05-25: First Release

978-1-491-92561-4

[TI]

Preface

With the reinvigoration of neural networks in the 2000s, deep learning has become an extremely active area of research that is paving the way for modern machine learning. This book uses exposition and examples to help you understand major concepts in this complicated field. Large companies such as Google, Microsoft, and Facebook have taken notice and are actively growing in-house deep learning teams. For the rest of us, deep learning is still a pretty complex and difficult subject to grasp. Research papers are filled to the brim with jargon, and scattered online tutorials do little to help build a strong intuition for why and how deep learning practitioners approach problems. Our goal is to bridge this gap.

Prerequisites and Objectives

This booked is aimed an audience with a basic operating understanding of calculus, matrices, and Python programming. Approaching this material without this background is possible, but likely to be more challenging. Background in linear algebra may also be helpful in navigating certain sections of mathematical exposition.

By the end of the book, we hope that our readers will be left with an intuition for how to approach problems using deep learning, the historical context for modern deep learning approaches, and a familiarity with implementing deep learning algorithms using the TensorFlow open source library.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/darksigma/Fundamentals-of-Deep-Learning-Book*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Fundamentals of Deep Learning* by Nikhil Buduma and Nicholas Locascio (O'Reilly). Copyright 2017 Nikhil Buduma and Nicholas Locascio, 978-1-491-92561-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgements

We'd like to thank several people who have been instrumental in the completion of this text. We'd like to start by acknowledging Mostafa Samir and Surya Bhupatiraju, who contributed heavily to the content of Chapters 7 and 8. We also appreciate the contributions of Mohamed (Hassan) Kane and Anish Athalye, who worked on early versions of the code examples in this book's Github repository.

This book would not have been possible without the never-ending support and expertise of our editor, Shannon Cutt. We'd also like to appreciate the commentary provided by our reviewers, Isaac Hodes, David Andrzejewski, and Aaron Schumacher, who provided thoughtful, in-depth commentary on the original drafts of the text. Finally, we are thankful for all of the insight provided by our friends and family

members, including Jeff Dean, Nithin Buduma, Venkat Buduma, and William, Jack, as we finalized the manuscript of the text.

Table of Contents

CHAPTER 1

The Neural Network

Building Intelligent Machines

The brain is the most incredible organ in the human body. It dictates the way we perceive every sight, sound, smell, taste, and touch. It enables us to store memories, experience emotions, and even dream. Without it, we would be primitive organisms, incapable of anything other than the simplest of reflexes. The brain is, inherently, what makes us intelligent.

The infant brain only weighs a single pound, but somehow it solves problems that even our biggest, most powerful supercomputers find impossible. Within a matter of months after birth, infants can recognize the faces of their parents, discern discrete objects from their backgrounds, and even tell apart voices. Within a year, they've already developed an intuition for natural physics, can track objects even when they become partially or completely blocked, and can associate sounds with specific meanings. And by early childhood, they have a sophisticated understanding of grammar and thousands of words in their vocabularies.[1]

For decades, we've dreamed of building intelligent machines with brains like ours—robotic assistants to clean our homes, cars that drive themselves, microscopes that automatically detect diseases. But building these artificially intelligent machines requires us to solve some of the most complex computational problems we have ever grappled with; problems that our brains can already solve in a manner of microseconds. To tackle these problems, we'll have to develop a radically different way of programming a computer using techniques largely developed over the past decade. This

1 Kuhn, Deanna, et al. *Handbook of Child Psychology. Vol. 2, Cognition, Perception, and Language*. Wiley, 1998.

is an extremely active field of artificial computer intelligence often referred to as *deep learning*.

The Limits of Traditional Computer Programs

Why exactly are certain problems so difficult for computers to solve? Well, it turns out that traditional computer programs are designed to be very good at two things: 1) performing arithmetic really fast and 2) explicitly following a list of instructions. So if you want to do some heavy financial number crunching, you're in luck. Traditional computer programs can do the trick. But let's say we want to do something slightly more interesting, like write a program to automatically read someone's handwriting. Figure 1-1 will serve as a starting point.

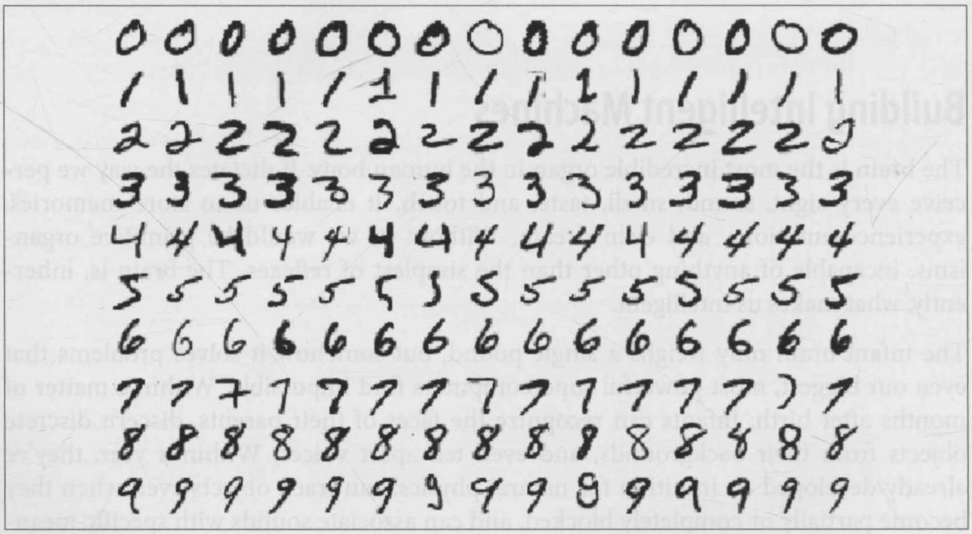

Figure 1-1. Image from MNIST handwritten digit dataset[2]

Although every digit in Figure 1-1 is written in a slightly different way, we can easily recognize every digit in the first row as a zero, every digit in the second row as a one, etc. Let's try to write a computer program to crack this task. What rules could we use to tell one digit from another?

Well, we can start simple! For example, we might state that we have a zero if our image only has a single, closed loop. All the examples in Figure 1-1 seem to fit this bill, but this isn't really a sufficient condition. What if someone doesn't perfectly close

2 Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition" *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

the loop on their zero? And, as in Figure 1-2, how do you distinguish a messy zero from a six?

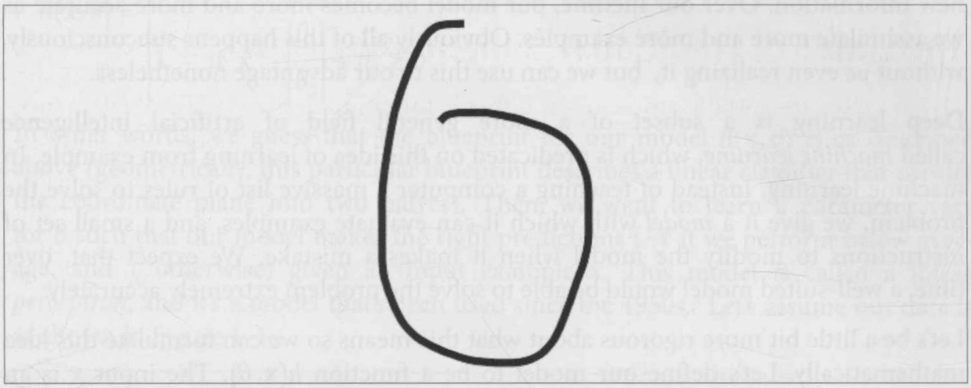

Figure 1-2. A zero that's algorithmically difficult to distinguish from a six

You could potentially establish some sort of cutoff for the distance between the starting point of the loop and the ending point, but it's not exactly clear where we should be drawing the line. But this dilemma is only the beginning of our worries. How do we distinguish between threes and fives? Or between fours and nines? We can add more and more rules, or *features*, through careful observation and months of trial and error, but it's quite clear that this isn't going to be an easy process.

Many other classes of problems fall into this same category: object recognition, speech comprehension, automated translation, etc. We don't know what program to write because we don't know how it's done by our brains. And even if we did know how to do it, the program might be horrendously complicated.

The Mechanics of Machine Learning

To tackle these classes of problems, we'll have to use a very different kind of approach. A lot of the things we learn in school growing up have a lot in common with traditional computer programs. We learn how to multiply numbers, solve equations, and take derivatives by internalizing a set of instructions. But the things we learn at an extremely early age, the things we find most natural, are learned by example, not by formula.

For instance, when we were two years old, our parents didn't teach us how to recognize a dog by measuring the shape of its nose or the contours of its body. We learned to recognize a dog by being shown multiple examples and being corrected when we made the wrong guess. In other words, when we were born, our brains provided us with a model that described how we would be able to see the world. As we grew up, that model would take in our sensory inputs and make a guess about what we were experiencing. If that guess was confirmed by our parents, our model would be reinforced. If our parents said we were wrong, we'd modify our model to incorporate this new information. Over our lifetime, our model becomes more and more accurate as we assimilate more and more examples. Obviously all of this happens subconsciously, without us even realizing it, but we can use this to our advantage nonetheless.

Deep learning is a subset of a more general field of artificial intelligence called *machine learning*, which is predicated on this idea of learning from example. In machine learning, instead of teaching a computer a massive list of rules to solve the problem, we give it a *model* with which it can evaluate examples, and a small set of instructions to modify the model when it makes a mistake. We expect that, over time, a well-suited model would be able to solve the problem extremely accurately.

Let's be a little bit more rigorous about what this means so we can formulate this idea mathematically. Let's define our model to be a function $h(\mathbf{x}, \theta)$. The input $\mathbf{x}$ is an example expressed in vector form. For example, if $\mathbf{x}$ were a grayscale image, the vector's components would be pixel intensities at each position, as shown in Figure 1-3.

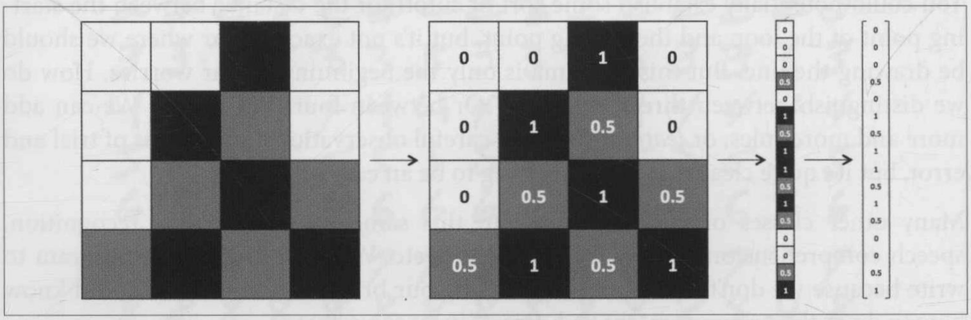

Figure 1-3. The process of vectorizing an image for a machine learning algorithm

The input θ is a vector of the parameters that our model uses. Our machine learning program tries to perfect the values of these parameters as it is exposed to more and more examples. We'll see this in action and in more detail in Chapter 2.

To develop a more intuitive understanding for machine learning models, let's walk through a quick example. Let's say we wanted to determine how to predict exam performance based on the number of hours of sleep we get and the number of hours we study the previous day. We collect a lot of data, and for each data point $\mathbf{x} = \left[x_1 \ x_2\right]^T$, we record the number of hours of sleep we got (x_1), the number of hours we spent studying (x_2), and whether we performed above or below the class average. Our goal, then, might be to learn a model $h(\mathbf{x}, \theta)$ with parameter vector $\theta = \left[\theta_0 \ \theta_1 \ \theta_2\right]^T$ such that:

$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

In other words, we guess that the blueprint for our model $h(\mathbf{x}, \theta)$ is as described above (geometrically, this particular blueprint describes a linear classifier that divides the coordinate plane into two halves). Then, we want to learn a parameter vector θ such that our model makes the right predictions (−1 if we perform below average, and 1 otherwise) given an input example **x**. This model is called a linear *perceptron*, and it's a model that's been used since the 1950s.[3] Let's assume our data is as shown in Figure 1-4.

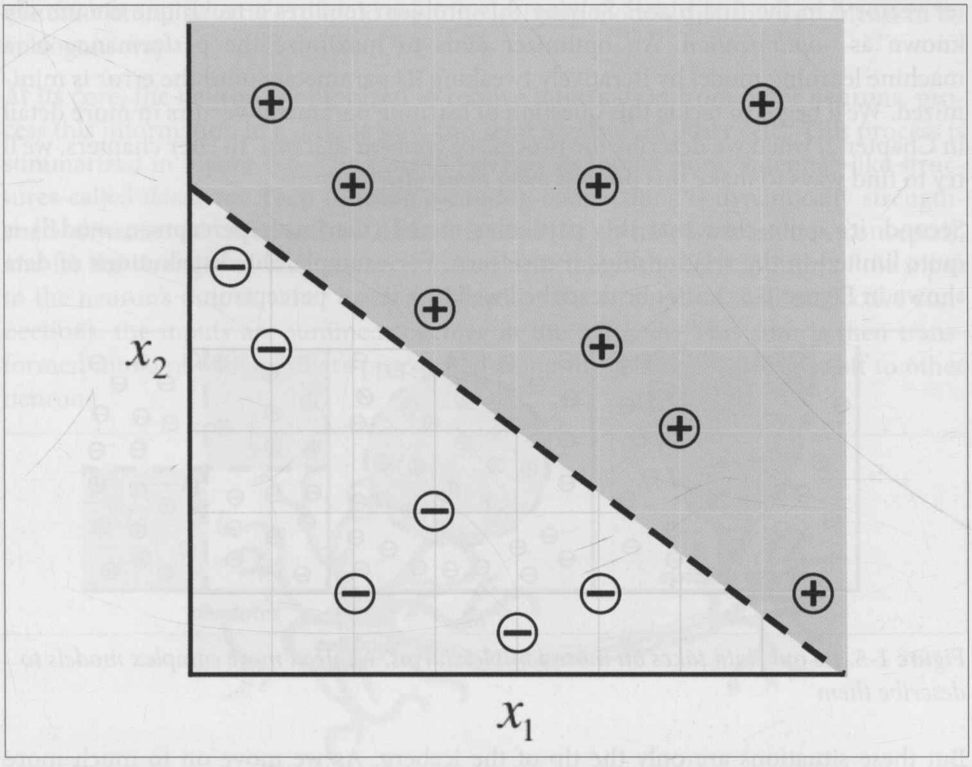

Figure 1-4. Sample data for our exam predictor algorithm and a potential classifier

3 Rosenblatt, Frank. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review* 65.6 (1958): 386.