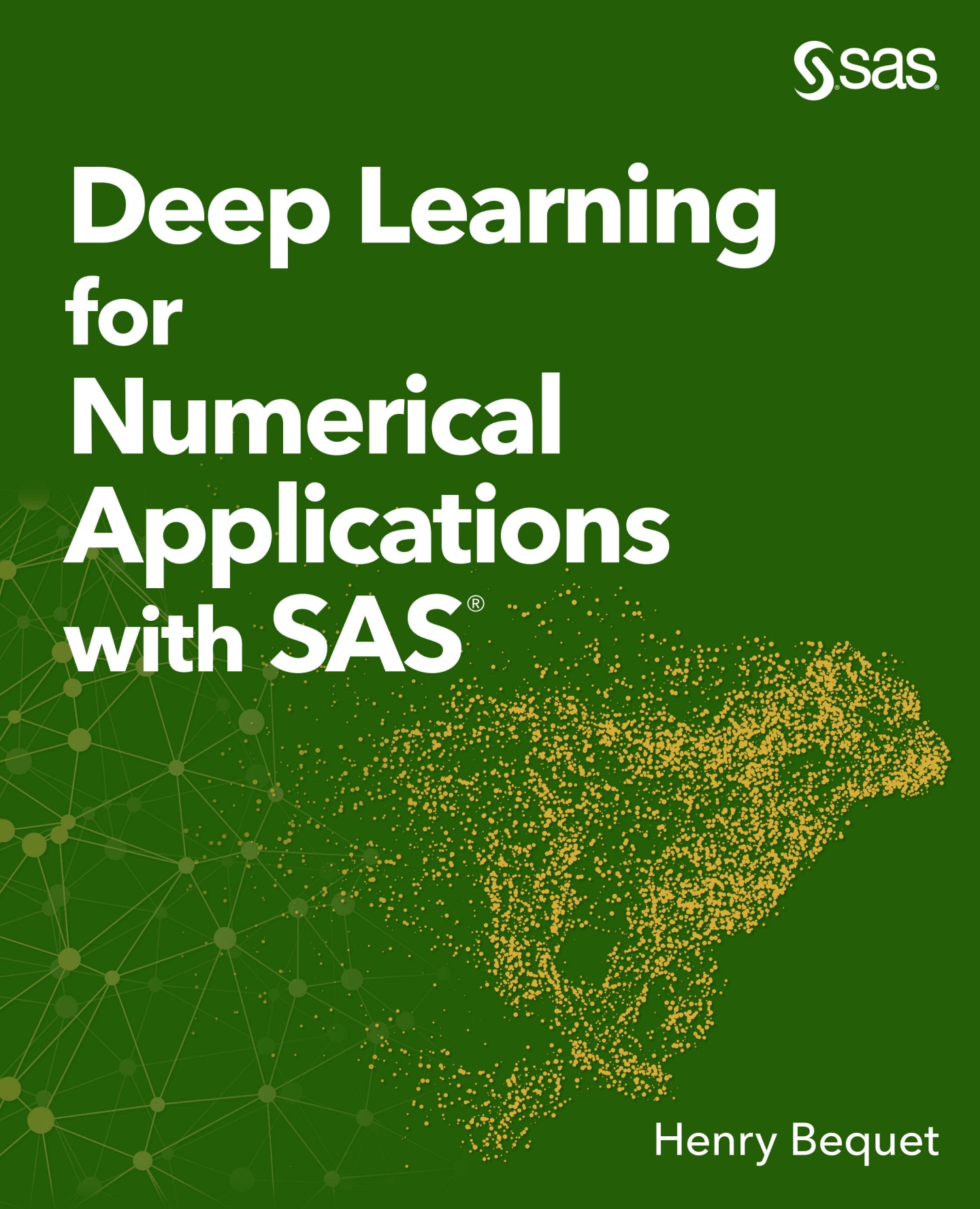
sas

Deep Learning for Numerical Applications with SAS®



Henry Bequet

The correct bibliographic citation for this manual is as follows: Bequet, Henry G. 2018. *Deep Learning for Numerical Applications with SAS®*. Cary, NC: SAS Institute Inc.

Deep Learning for Numerical Applications with SAS®

ISBN 978-1-63526-680-1 (Hardcopy)
ISBN 978-1-63526-677-1 (EPUB)
ISBN 978-1-63526-678-8 (MOBI)
ISBN 978-1-63526-679-5 (PDF)

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

July 2018

Contents

Preface

Artificial Intelligence (AI) and Machine Learning (ML) are all the rage. Computerized systems that can perform human tasks and make decisions are affecting many industries.

A core technology of these systems is deep learning, which is based on deep neural networks. Neural networks are not new, yet the successes in artificial intelligence are relatively recent. The availability of more computing power through multicore CPUs and Graphics Processing Units (GPUs) enabled us to train deeper networks. The availability of big data enabled us to train these networks well. The availability of specific neural networks—such as convolutional and recurrent networks—fueled the advances in image processing and natural language processing.

Combined, these forces created the perfect substrate for AI applications to grow.

Henry Bequet reminds us in this book that neural networks are algorithms to predict outcomes, to classify observations, and to detect patterns. They have many applications outside of computer vision, chatbots, and autonomous vehicles. The forces that accelerated progress through deep learning in cognitive analytics can be brought to bear in other domains, such as regression, function approximation, and Monte Carlo simulation.

In this book, Henry takes you on a tour of deep learning with SAS® using surprising applications that broaden your understanding of the technology. Henry guides you through the deep learning capabilities of SAS® Viya® that extend and complement your SAS experience.

Oliver Schabenberger, PhD
Executive Vice President, Chief Operating Officer and Chief Technology Officer
SAS

About This Book

What Does This Book Cover?

Machine learning and deep learning are ubiquitous in our homes and workplaces, from machine translation, to image recognition, to predictive analytics, to autonomous driving. Deep learning holds the promise of improving many of the applications that we use every day in a variety of fields. Most of the deep learning literature that is currently available explains the mechanics of deep learning with the goal of implementing cognitive applications fueled by big data. This book is different. Written by an expert in high-performance analytics, this book introduces a new field: deep learning for numerical applications (DL4NA). In contrast to deep learning, the primary goal of DL4NA is not to learn from data. The primary goal of DL4NA is to dramatically improve the performance of numerical applications by training deep neural networks.

This book presents the concepts and techniques step by step in a practical way so that you can easily reproduce the examples on your high-performance analytics systems. This book also discusses the latest hardware innovations that can power your SAS programs, including many-core CPUs, graphics processing units (GPU), field-programmable gate arrays (FPGA), and application-specific integrated circuits (ASIC).

Is This Book for You?

This book assumes no prior knowledge of high-performance computing, machine learning, or deep learning. It is for SAS developers and programmers who want to develop and run the fastest analytics.

It is also for those who are curious about the roots of deep learning and want an introduction to this fascinating field.

What Are the Prerequisites for This Book?

The prerequisites of this book are familiarity with SAS and the SAS programming language.

What Should You Know about the Examples?

This book includes tutorials for you to follow to gain hands-on experience with SAS.

Software Used to Develop the Book's Content

SAS 9.4 M5 (including SAS Studio)

SAS Viya 3.3

SAS Infrastructure for Risk Management 3.4

Example Code and Data

You can access the example code and data for this book by linking to its author page at https://support.sas.com/bequet. Larger versions of some flow diagrams are also available on this page.

We Want to Hear from You

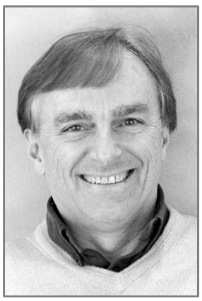
SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books to do the following:

- Sign up to review a book
- Recommend a topic
- Request information on how to become a SAS Press author
- Provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com or https://support.sas.com/author_feedback.

SAS has many resources to help you find answers and expand your knowledge. If you need additional help, see our list of resources: sas.com/books.

About The Author



Henry Bequet is Director of High Performance Computing and Machine Learning at SAS. In that capacity, he leads the development of a high-performance solution that can run SAS code on thousands of CPU and GPU cores for advanced models that use techniques like Black-Scholes, Binomial Evaluation, and Monte-Carlo simulations. Henry has more than 35 years of industry experience and 15 years of high-performance analytics practice. He has published two books and several papers on server development and machine learning.

Learn more about this author by visiting his author page at http://support.sas.com/bequet. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Chapter 1: Introduction

Deep Learning

This is a book about deep learning, but it is not a book about artificial intelligence.

In the remainder of this introduction, we explain those two statements in detail with a simple goal in mind: to help you determine whether this book is for you.

Let's begin by briefly discussing deep learning (DL)—more specifically, its pros, cons, and applicability. Then we will discuss the main motivation of this book: execution speed of analytics. We will defer a discussion on the mechanics of DL to Chapter 2.

For our discussion, we view DL as a technology with a straightforward goal:

Build a system that can predict outputs based on a set of inputs by learning from data.

You will notice that there are absolutely no references to a human brain, cognitive science, or creating a model of human behavior in this book. DL can do all those things and can do them very well, but that is not the focus of this book. For this book, we simply concentrate on creating a model (or building a system) that can predict outputs with some level of accuracy, given some inputs.

Like many technologies (some might argue any technology), DL has its advantages and disadvantages. Let's start with the advantages to keep our motivation high in these early stages.

Here are three of the main advantages of DL:

- DL provides the best performance on many data-driven problems. In other words, DL provides the best accuracy and the fastest results. That is a bold claim that has been proven mathematically in some cases and empirically in many others. We investigate this bold claim in more detail in Chapter 3.
- DL provides great model and performance portability. A DL network developed for one problem can often be applied to many other problems without a significant loss of accuracy and performance. We see vivid examples of this portability in Chapters 3 and 7.

- DL provides a high level of automation of your model. Someone with good DL skills but little domain knowledge can easily create state-of-the-art models. Chapter 7 illustrates how powerful that characteristic is for modeling random walks.

These key advantages come at a cost:

- DL is computational and data intensive. Without a lot of both computational power and data, the accuracy of your DL models will suffer to the point of not being competitive.
- DL will not give out its secrets. This is true during training, where specifying the correct parameters is an art more than a science. This is also true during inference (a term that we define more clearly in Chapter 2). As you might already know and as we will show you in the remainder of this book, DL can give you great predictive accuracy for your models, but you cannot completely explain why it works so well.

Both of those disadvantages can be crippling, so let's discuss them further to help you determine their impact on your problems.

Is Deep Learning for You?

Computing resources during training was a crippling factor for neural networks during the last decade of the 20th century: the computing power wasn't available to train any but the simplest networks. Note that the term "deep learning" hadn't been coined yet; it most likely originates from the reviews and commentary of Hinton et al. (2006). The availability of computing resources is becoming less of a problem today thanks to the advent of many-core machines, graphics processing unit (GPU) accelerators, and even hardware specialized for DL.

Why is DL hungry for computing resources? Simply put, it's because DL is a subfield of computer science, and computer science thrives on computational resources. Without access to a lot of computational resources, you will not do well with DL. How much is a lot? Well, it depends, and we give some guidelines in quantifying computing resources in Chapters 4 and 8.

The fact that DL requires a lot of data for training is significant if you don't have the data. For example, if you're trying to predict shoppers' behaviors on an e-commerce website, you are likely to fail without accurate data. Manufacturing the data won't help in this case, since you are trying to learn from the data. Note that having an algorithm to manufacture data is a good sign that you understand the data. There are many other examples where a lot of data has made things possible and the absence of data is a crippling obstacle (Ng 2016).

The examples that we use in this book don't suffer from this drawback. When we don't have the data, we can manufacture it. For example, if we are trying to improve upon Monte Carlo simulations, as we do in Chapter 5, and we discover that we need a larger training set, there is nothing to worry about. We can simply run more Monte Carlo simulations to generate (manufacture) more data. In Chapter 6, we introduce one of the most powerful tools in the arsenal of the data scientist to produce a lot of training data: the general purpose graphics processing unit (GPGPU), or simply the graphics processing unit (GPU).

It's All about Performance

In the remainder of this introduction, we focus on speed, which is the main focus of this book. By now you must have decided that you can live with the drawbacks of DL that we just discussed. So you have enough data, have plenty of computing resources, and can live with the black box effect (the fact that DL doesn't give out its secrets) that often worries statisticians (Knight 2017).

If you're still on the fence, maybe the performance argument will convince you one way or another.

Flynn's Taxonomy

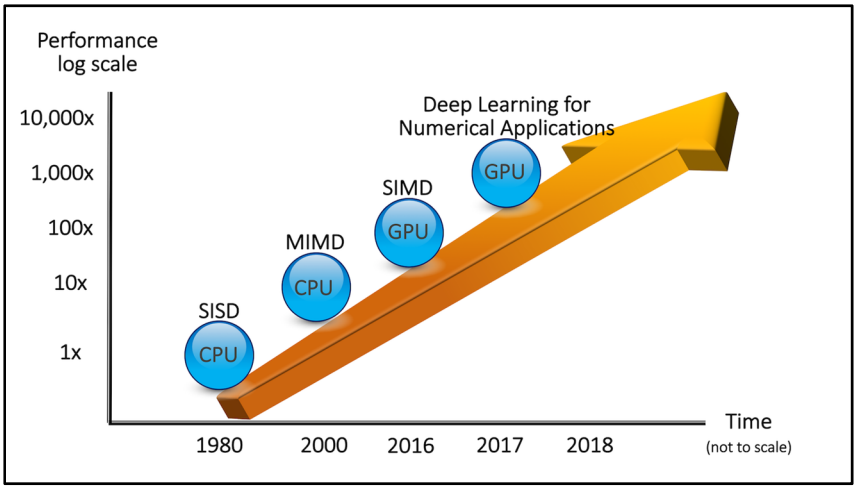
Most of the work presented in this book finds its roots in the Financial Risk Division at SAS. Financial institutions use a large number of computations to evaluate portfolios, price securities, and financial derivatives. Time is usually of the essence when it comes to financial transactions, so having access to the fastest possible technology to perform financial computations with enough accuracy is often paramount.

To organize our thinking around numerical application performance, let's rely on the following categories from Flynn's taxonomy (Flynn 1972):

- **Single instruction, single data (SISD)**
  A sequential computer that exploits no parallelism in either the instruction or data streams.
- **Multiple instruction streams, multiple data streams (MIMD)**
  Multiple autonomous processors simultaneously executing different instructions on different data.
- **Single instruction stream, multiple data streams (SIMD)**
  A computer that exploits multiple data streams against a single stream to perform operations that might be naturally parallelized.

Figure 1.1 shows Flynn's taxonomy on a timeline with the technologies associated with each classification (for example, GPUs are for SIMD). The dates and performance factors in Figure 1.1 are approximate; the main point is to give the reader an idea of the performance improvements that can be obtained by moving from one technology to another. As you will see as you read this book further, the numbers in Figure 1.1 are impressive, yet very conservative.

Figure 1.1: Performance of Analytics



Life after Flynn

We start our exploration of the performance of numerical applications around 1980, when systems such as SAS started to be widely used. The SAS system (`sas.exe` that still exists today) is a SISD engine: SAS runs analytics one operation at a time on one data element at a time. Over the years, multi-threaded functionality has been added to SAS (for example, in PROC SORT), but at its heart SAS remains a SISD engine.

From the year 2000 to 2015 or so, analytics started to go MIMD with multiple cores and even multiple machines. Systems such as the SAS Threaded Kernel, the SAS Grid, Map-Reduce, and others gave folks access to much improved performance. We chose to give MIMD a 10x in our chart, but its performance was often much greater.

MIMD systems had and still have two main challenges:

- Make it as easy as possible to distribute the work across multiple cores and multiple machines.
- Keep the communication between the cores and the machines as light as possible.

As of this writing, finding good solutions to those two challenges still consumes a great deal of energy in the industry, and new products are still introduced, such as SAS Viya and SAS Infrastructure for Risk Management, to name only a couple. In terms of performance, the progress being made in the MIMD world is incremental at this point, so to go an order of magnitude faster, a paradigm shift is needed.

That paradigm shift comes in the form of the general purpose graphics processing unit (GPGPU), or simply graphics processing unit (GPU). GPUs are SIMD processors, so they need SIMD algorithms to process. To run quickly on GPUs, many algorithms have been redesigned to be implemented as SIMD algorithms (Satish et al. 2008). For example, at the time of this writing, most problems that occupy financial risk departments have a SIMD implementation. The most notable counter-examples are reports and spreadsheets. Potentially every single cell in a spreadsheet or a report implements a different formula (algorithm). This makes the whole report or spreadsheet ill-suited for SIMD implementations.

This last observation about reports and spreadsheets brings up an important point: as one moves up in our chart in Figure 1.1, not all problems can be fitted into the upper bubbles. Roughly speaking, any computable problem can be implemented with a SISD algorithm, a clear majority of the computable problems can be implemented with a MIMD algorithm, and a great number of problems can be implemented with a SIMD algorithm. One could visualize this applicability of algorithms to problems as an inverted cone. At the top of the cone (in the wide part), you find all applications that run on a computer, including yours. As you move down the cone, the number of applications shrinks, but at the same time the performance goes up. In other words, the closer to the bottom of the cone, the faster your application, but the less likely you are to find your application. As time goes by and new algorithms are developed, the narrow (bottom) tip of the cone becomes wider and wider.

But SIMD is not the final answer to fast performance for analytics; it is the beginning of the endeavor that we describe in this book.

We believe that the next paradigm shift with respect to the performance of numerical applications will come from deep learning. Once a DL network is trained to compute analytics, using that DL network becomes dramatically faster than more classic methodologies like Monte Carlo simulations. This latest paradigm shift is the main topic of this book.

Organization of This Book

This is a practical book: we want you to be able to reproduce the sample on your hardware with Base SAS and SAS Studio. You will not get the same results as what we publish in the book if you don't have the same hardware as what we used (who knows, yours might be faster!), but you will obtain similar results. To get the most out of the book, we advise you to follow the examples along with the book.

In Chapter 2, "Deep Learning," we provide a practical introduction to DL by describing the Deep Learning Toolkit (TKDL) that is available to SAS users. We start with a simple example of a cognitive application and then discuss how DL can go beyond cognitive applications.

Going beyond cognitive applications is precisely what we will do in Chapter 3, "Regressions." In that chapter, we show how the reader can use SAS in an application of the universal approximation theorem.

In Chapter 4, "Many-Task Computing," we take a slight digression from DL into supercomputing to introduce scalable deep learning techniques. In this chapter, we also discuss data object pooling, a technique that high-performance computing uses more and more to dramatically accelerate daily analytics computations. Chapter 4 provides one of the pillars of the foundation of the rest of book (the other pillar is DL).

In Chapter 5, we study Monte Carlo simulations. We begin with a simple deterministic example and then we progress to a stochastic problem.

In Chapter 6, "GPU," we leverage the awesome SIMD power of GPUs to manufacture extensive training data for a DL network.

In Chapter 7, "Monte Carlo Simulations with Deep Learning," we study how Monte Carlo simulations can be approximated using DL. The main takeaway from this chapter is that with a limited understanding of a domain and good DL skills, you can implement state-of-the-art analytics, both in terms of accuracy and in terms of performance.

In Chapter 8, "Deep Learning for Numerical Applications in the Enterprise," we describe how to gradually introduce deep learning for numerical applications into enterprise solutions. The main goal of this chapter is to convince you that the technologies described so far can be used to introduce an evolution to deep learning for numerical applications, not a revolution. We also discuss the best practices and pitfalls of scalability for deep learning.

Finally, in Chapter 9, "Conclusions," we summarize why deep learning for numerical applications is a powerful technique that allows SAS users to marry traditional analytics and deep learning to their existing analytics infrastructure. We also briefly discuss specialized hardware that will quickly become a viable solution because of the universality of DL.

But let's not get ahead of ourselves; we first need to look at the basics of DL and how to implement DL with SAS.