The first guide to Prediction AP

SAMPLE

BOOTSTRAPPING MACHINE LEARNING

Exploit the value of your data. Create smarter apps and businesses.

By Louis Dorard, Ph.D.

Bootstrapping Machine Learning

Website

www.louisdorard.com/machine-learning-book

Release

Version 1.0.5 — 8 April 2014

Written by

Louis Dorard

THANKS!

Thank you for downloading this sample of *Bootstrapping Machine Learning*! I hope you'll enjoy it.

If you have any thoughts to share, please email me at louis@dorard.me and I will reply to you personally.

Louis

CONTENTS

CHAPTER 1: INTRODUCTION

1

WHO IS THIS BOOK FOR?

This book is for computer engineers, scientists, hackers, programmers, analysts, and thinkers. We are building businesses and applications, for ourselves or for others. Despite coming from different backgrounds, part of our jobs is to process information. We collect it, we move it from one place to another, and we store it as *data*. The next step after gathering this data is to use our machines to analyze it, learn from it, and create new information based on what they have learned.

This new information might be answers to specific questions we've asked about the data, such as "How much?" and "Which one?" For example, you might record real estate transaction prices and ask the price of a property that has not been sold yet. Or you may have received comments to a blog post and would like to identify which are spam or offensive, to filter them out. Because the answers to these questions are not to be found directly in the data, we call them predictions.

The ability to make such predictions holds a lot of value. When it's built into applications to make them smarter, the users have an improved experience. Observing a user's behavior and gathering data from his interactions with the app can be used to understand his interests and predict his next move. It's like having personalized virtual assistants for each user who work extremely fast to make the app more engaging. Predictions can also be used to improve a business by analyzing and making sense of customer data. One example of that is predicting which customers are likely to cancel a service subscription, and taking appropriate action to keep them from leaving. In all of these examples, making predictions from data results in increased revenue.

Up until a few years ago, the science of making such predictions by having a machine "learn" from data required a strong background in mathematics, probability, statistics, computer science, and a lot of practical experience. Your only chance was to hire experts in the field, or be one yourself. Luckily, the 2010s have seen a major technological advance through the introduction of new Prediction Application Programming Interfaces on the web (Prediction APIs). Prediction APIs can be used with no prior knowledge of data science. Indeed, they simply consist in two methods: one method to feed data, and one method to ask questions and get predictions. My decision to write this book was inspired by everything Prediction APIs can offer.

You will find many textbooks on the market to teach you the inner workings of the Machine Learning algorithms and of the Data Mining techniques that are used to make data-based predictions. They will teach you everything from the theory behind these algorithms to how to implement them and how to apply them to real-world problems. My book is not one of them.

My intent is to cover only the basic concepts so you can start making predictions from data and bootstrap the use of Machine Learning in your projects, applications, or business. I will teach you just enough for you to use Prediction APIs effectively. I will explain how to formulate your problem as a Machine Learning problem, and what the algorithms behind the APIs are supposed to do — not how they work. If you ever decide in the end that you want to become an expert in Machine Learning, I believe you will be in a much better position to do so after reading this book.

I am writing this book to help you make better applications and businesses. I want to show you that these days, everyone is in a position to exploit the power of Machine Learning with minimal coding experience and the use of Prediction APIs. It doesn't take a Master′s degree and it doesn't require a big budget. You will actually save a lot of money by not hiring an expert and you will even have better chances to incorporate your domain knowledge by handling things yourself.

I am writing this book for bootstrappers who understand why you should do a job yourself before hiring someone else to do it. I am writing for Über-builders who know how important it is to master all the aspects of a product. I am writing for developers who want to extend their skill-set and for thinkers who want to broaden their horizons. I am writing to help you succeed and differentiate yourself by using Machine Learning.

STRUCTURE OF THIS BOOK

Let’s start our journey into the world of Machine Learning and Prediction APIs with some prerequisites. First, I'll situate ML within what we call Artificial Intelligence and other techniques used to make computers "intelligent." I’ll continue with some basic concepts and terminology so we can characterize what ML does more precisely and introduce the Prediction APIs that give access to ML as a Service. ML is about learning from examples, so I’ll give you examples of ML uses to help you learn the concepts of what ML does. If things ever start to get too abstract for you while reading this book, you can refer to these examples for context.

The rest of the book will focus on guiding you to put Prediction APIs to use. We will see how to formulate your problem as an ML problem, how to prepare your data, and how to integrate predictions in your application. We will then review Prediction APIs and see how to use them in more detail. Be ready to get your hands dirty after that, because we'll be applying everything we said and we’ll be using Prediction APIs on a concrete case.

Happy reading!

CHAPTER 2: MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

2

MOTIVATIONS

Computers are really good at processing lots of information. Think about the millions of messages flowing to the Twitter servers, which then have to figure out trending topics in real time. But they lack intelligence. They need to be programmed with precise lists of basic instructions to perform. This makes it challenging for them to do certain things that humans do very easily, but for which we do not know the exact process. For example, think about recognizing hand-written digits: can you explain to me, step-by-step, how you can recognize numbers that form the ZIP code on a hand-written address?

There are diverse techniques to make computers do "intelligent" or "human-like" things, and they form the field of Artificial Intelligence (AI). We'll have a word of caution, first, on this terminology. "Intelligence" and "human-like" do not mean that computers think and have the same mental capabilities as us humans. It just means that they do things that are non-trivial and that only we could do before, such as playing chess. The AI techniques used in chess programs work by examining possible sequences of moves (player A plays this move, then player B may play these moves, then player A may play those moves, etc.) and by evaluating who has the advantage at the end of each of these sequences, simply by counting who has more pieces on the board. There's not much intelligence in that technique, but it does the job! Chess is an example of something that's easy for computers but hard for humans, but it doesn't mean that computers are more intelligent than us. Another example is recognizing people in video footage: this is *very hard* for computers but dead easy for humans.

Machine Learning (ML) is a set of AI techniques where intelligence is built by referring to examples. In the hand-written digit recognition problem, you would use ML by providing examples of hand-written digits (in the form of scanned images) along with their corresponding integers between 0 and 9. These examples are called the "data." When given a new image of a hand-written digit that's never been seen before, the idea is to compare it to the example images and make a prediction based on the integers that correspond to the examples most similar to the new image. This is very similar to what a human would do; Machine Learning is about making machines reproduce how humans learn.

If computers can be just as smart as a human being on a given task, the immediate benefit is that this task may be performed with greater speed and at lower costs. For certain tasks, our own human reasoning may be, just as in Machine Learning, to compare to examples — or in other words, refer to our own previous experience. In this case, we might even expect computers to outsmart us, thanks to their ability to process more information.

An example of referring to experience is establishing diagnoses based on patients' symptoms. Doctors refer to their own experience or to the collective experience that has been synthesized in the literature. They refer to examples of previous patients for whom a diagnosis was made, and they compare new patients to them. The hope with computers is that their processing power would allow them to consider many more previous patients, but also be able to consider more information about each patient (demographic, diet, clinical measurements, and medical history). Doing so should improve the overall accuracy of predictions.

Machine Learning is a set of AI techniques where intelligence is built by referring to examples.

ML-POWERED APPS

Let me now give a few examples of how ML is used in popular web and mobile apps.

On the web, giant tech companies such as Google, Yahoo and Microsoft use Machine Learning extensively to perform tasks that require intelligence on a huge scale. They are able to show personalized content to each of the hundreds of millions of visitors of their web properties by predicting what is most likely to engage each user. Every time a user interacts with their website, it gives them example data of what content interests which user. This content can be articles on a news site homepage or ads on a search engine results page.

Google Mail (Gmail) is an interesting example of a web application which uses Machine Learning to offer killer new features to its users. With Priority Inbox, Google have taken email categorization a step further than traditional email clients who are able to identify spam emails: Gmail can also identify important emails. I personally find it a huge time saver as it helps me filter the noise in my inbox and keep it clean. Besides, I can help the system learn what is important to *me* by marking or unmarking emails as important. Unlike desktop clients, Gmail is continuously learning and updating in real time with the data collected from all the users' interactions. As you can see, Machine

Learning opens up new ways for users to interact with web applications, with the promise of improving their experience. ML-powered features act as individual assistants to applications' users, by making intelligent predictions based on collected data.

On mobile, Apple has empowered the owners of their iDevices with an actual virtual assistant, Siri. Queries are made to it via voice. The user holds down a button, speaks, and waits for Siri to process the query and respond. The type of queries that Siri can process are making a call, sending a text, finding a business, getting directions, scheduling reminders and meetings, searching the web, and more. In Siri, ML is used all over the place for speech recognition (transforming the audio recording of the query to text), for recognizing the type of query among all those that can be processed, and for extracting the "parameters" of the query (e.g. for a query of the type "reminder", extract the name of the task and time to be reminded at — "remind me to do *a certain task* at *a certain time*"). Once this is done, Siri calls the appropriate service/API with the right parameters, and voilà! The aim here is to allow mobile users to do more with their devices, with less interaction. Often times, mobile usage situations make it more difficult to interact with the device than with a desktop machine. Using Machine Learning is a way to make it easier to interact with mobile apps.

The photo upload interface on the Facebook mobile app gives us an example of how ML can help improve the usage of an app. Immediately after selecting a photo, the app scans the image for faces, and little dialogs are positioned below every face that has been detected, prompting the user to tag the person whose face appears on the picture. This creates more user engagement and allows Facebook to gently remind its users that their experience is more fun if they tag friends on their pictures.

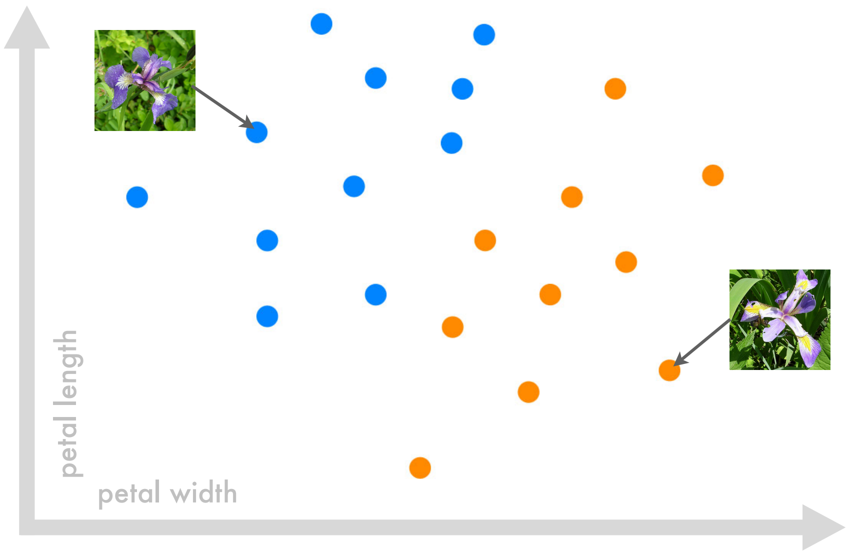
You will find these examples presented in more details in Chapter 4, but bear with me for a moment until I introduce the main concepts of Machine Learning and some terminology (Chapter 3).

ML-powered features act as individual assistants to applications' users, by making intelligent predictions based on collected user data.

A GRAPHICAL EXAMPLE

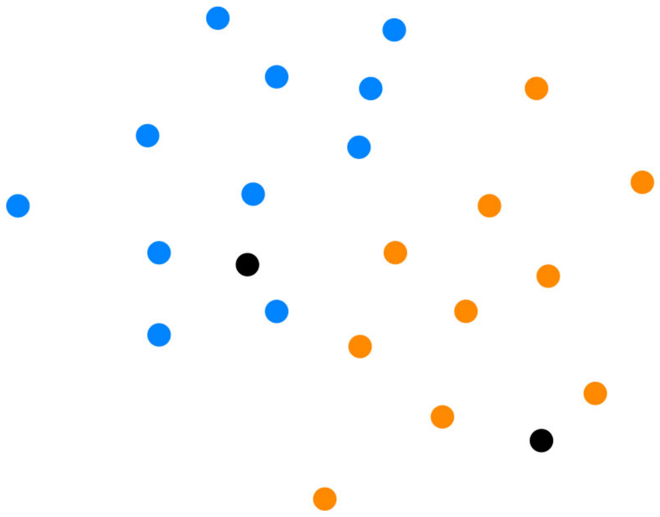
Let’s now have a look at how ML works with a graphical illustration of the intuition behind a categorization algorithm. Consider objects that are characterized by two attributes, and to which you want to assign one of two categories. For instance, let's say you want to categorize iris flower species (this is the "hello world" of ML). We’re not botanists, but we can measure petal lengths and widths to characterize the flowers.

Suppose we have collected example data, consisting of records of petal length and width of *setosa* and *virginica* iris flowers. We can plot these measurements as points on a 2D graph showing the petal length on the horizontal axis and the width on the vertical axis. The species information determines the color of each plotted point: orange for virginica and blue for setosa.

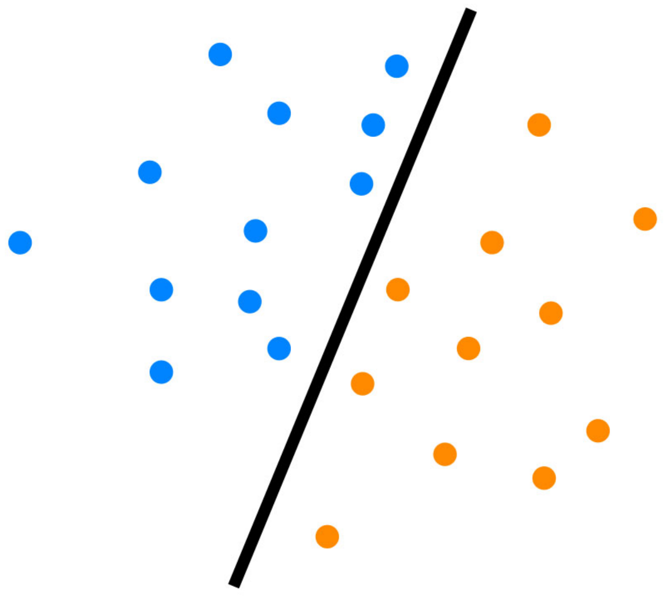


Now imagine that we're given new pairs of length-width petal measurements, but we're not given any information about the corresponding species. We want to predict the most likely species for each of the measurements. In ML-speak, we say that we're given new *inputs* (here the length-width pairs) and that we want to predict the most likely *output* for each of the inputs.

We'll plot the new inputs in black for the moment.

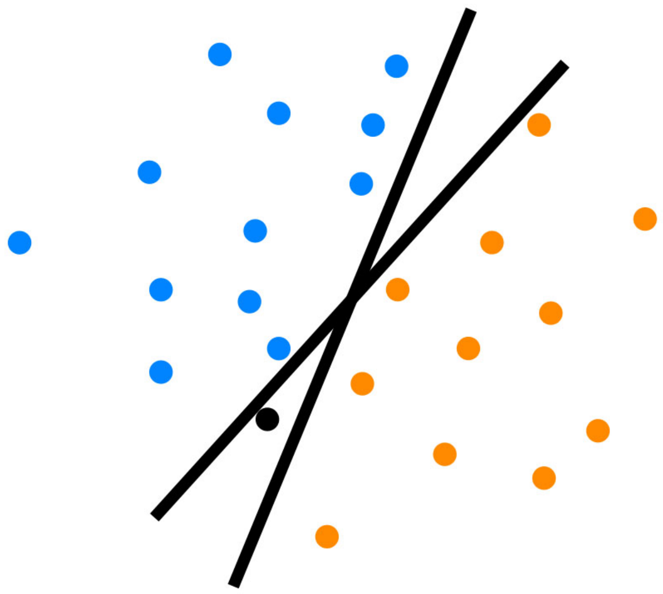


The color to assign to them, i.e., which predictions to make, looks fairly easy. The leftmost point should be blue and the rightmost point should be orange. So, let's just draw a line that separates the orange points from the blue points.

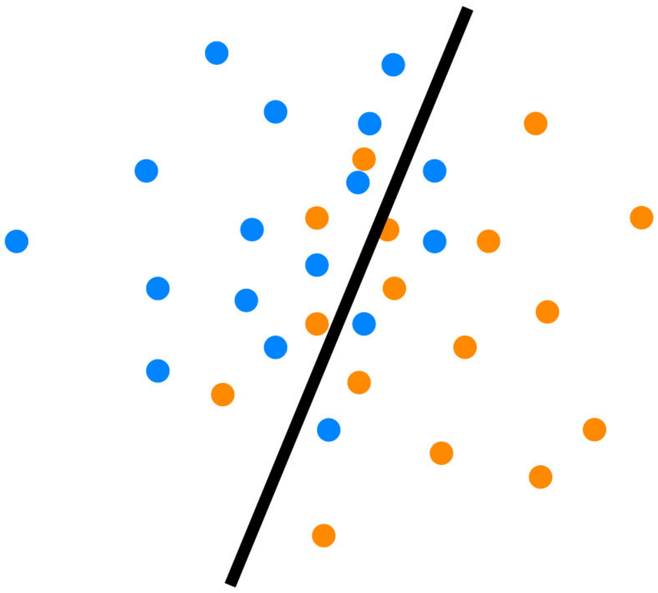


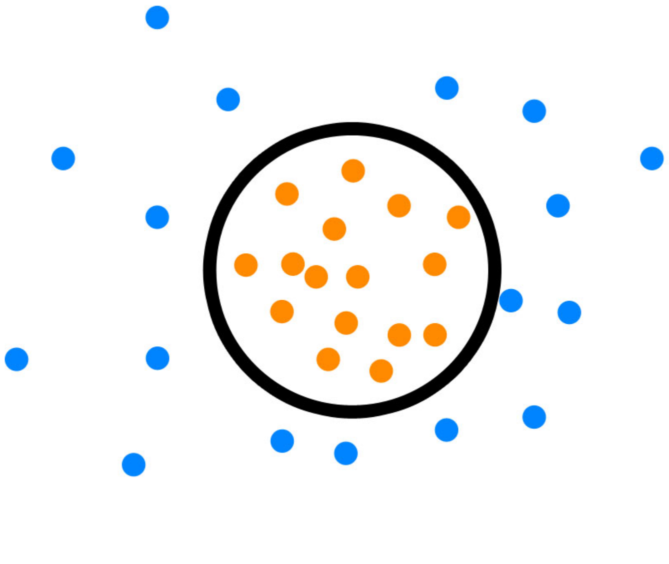
In the future we will affect colors based on which side of the black line the points fall on. That's it! That's our algorithm. What we're implicitly doing here is comparing new inputs to the example inputs we had, looking at the color (output) of those that are "in the same area." Our knowledge gathered from the examples that were given to us originally have taken the form of a line.

You will notice that there are several lines which would do the job of separating the orange examples from the blue examples. As you can see on the next figure, the line you choose to draw has an impact on some predictions, hence on the quality of predictions.



Machine Learning theory guides the choice of an optimal line. It would also allow us to find an optimal line of separation when the orange and blue populations, as plotted from the examples, overlap. And it can even find a non-linear boundary that would be better suited at separating the two populations.





As you'll see later, this has been implemented for us in Prediction APIs, so we don't need to worry about how the boundary for separating orange from blue was actually found. We just need to know that it is determined from the examples.

LEARNING VS HAND-CODING THE RULES

Before Machine Learning came about, the way we categorized objects was with a set of hard-coded rules, or, a combination of if-then-else statements. For instance, we would have said," If the petal width is smaller than w1 and if the length is smaller than l1, then this is a setosa flower. Otherwise, if the width is smaller than w2," ... and so on.

This sort of thing has been done in the field of language processing, which consists of extracting meaningful information from text or spoken words. Algorithms were based on complex sets of expertly hand-crafted rules that required very deep insights into the problems at hand. There were lots of parameters to consider, and you had to think about all the possible cases.

The use of ML makes predictions much easier than the rule-based systems. ML algorithms are flexible since they create their own "rules" for the problem at hand, based on the analysis of example inputs and corresponding outputs. There is no need to code any rule by hand — just to provide examples. In the previous illustration of the functioning of an ML algorithm for categorizing objects, a simple rule was created by learning a boundary between the example inputs in the two categories: if a new input is on this side of the boundary, it is from one category, and if it is on the other side, it is from the other category. In the case of a linear boundary and for an input represented by a (x1, x2) pair, the rule takes the form:

```
if (a*x1 + b*x2 + c >= 0) then category1;
else category2;
```

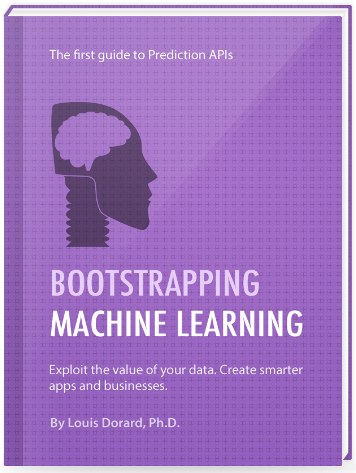
This is only one rule, but a very "smart" one since it applies to a quantity (a*x1 + b*x2 + c) which is calculated from a combination of the input representation and constants (a, b and c) derived from an analysis of the examples.

The late 1980s saw a revolution in language processing with the introduction of ML that led to important improvements over rule-based systems. Some of the earliest ML algorithms, called "decision trees," automatically produced systems of hard if-then rules similar to existing hand-written rules. The more recent algorithms do not follow the structure of decision trees, which were mostly motivated by the desire to make predictions based on decisions that remained easily interpretable by humans.

Do you find yourself writing rules by hand? If so, it's time to try for a little ML!

As a side note, whereas we illustrated an ML algorithm that is based on a geometric argument, it is worth knowing that there is a whole other class of algorithms based on probabilistic arguments. I won't go into any details on their functioning since it would require notions of probability theory. These algorithms are popular in language processing because they are generally more robust when given unfamiliar or erroneous input, as is very common in the real world. They also produce more reliable results when integrated into larger systems comprising multiple subtasks, since they allow a fine modeling of the uncertainty in the predictions.

*ML is flexible as **it automatically creates its own rules for the problem at hand,** based on an analysis of examples.*


The first guide to Prediction APIs
BOOTSTRAPPING
MACHINE LEARNING
Exploit the value of your data. Create smarter apps and businesses.
By Louis Dorard, Ph.D.

READ THE REST

Want to read the rest of this book? Interested in learning more about the possibilities of ML, the different types of ML problems, the APIs that exist to tackle them, how to use them for your problem, and in seeing it all in action in a case study?

Head over to
www.louisdorard.com/machine-learning-book
to order your full copy or one of the packages.

ABOUT LOUIS DORARD

I studied Machine Learning at University College London, where I obtained my Ph.D. At the same time, I also bootstrapped an online business.

In between that and writing this book, I organized a challenge and workshop on website optimization at the International Conference on Machine Learning, and I then served as Chief Science Officer in a startup where I was in charge of starting a Research & Development program and turning it into a product.

I love starting things from scratch and I am passionate about technological innovation in web and mobile apps. My experience has taught me the importance of simplicity and of starting small.

My goal now is to help people create smarter apps and businesses, and train them to use Prediction APIs. For those who need more personalized help, I have created a consultancy, Codole.

Follow me on Twitter for updates on what I am working on: @louisdorard.