1 Getting Started with Python Machine Learning

Machine learning teaches machines to learn to carry out tasks by themselves. It is that simple. The complexity comes with the details, and that is most likely the reason you are reading this book.

Maybe you have too much data and too little insight. You hope that using machine learning algorithms you can solve this challenge, so you started digging into the algorithms. But after some time you were puzzled: Which of the myriad of algorithms should you actually choose?

Alternatively, maybe you are in general interested in machine learning and for some time you have been reading blogs and articles about it. Everything seemed to be magic and cool, so you started your exploration and fed some toy data into a decision tree or a support vector machine. However, after you successfully applied it to some other data, you wondered: Was the whole setting right? Did you get the optimal results? And how do you know whether there are no better algorithms? Or whether your data was the right one?

Welcome to the club! Both of us (authors) were at those stages looking for information that tells the stories behind the theoretical textbooks about machine learning. It turned out that much of that information was "black art" not usually taught in standard text books. So in a sense, we wrote this book to our younger selves. A book that not only gives a quick introduction into machine learning, but also teaches lessons we learned along the way. We hope that it will also give you a smoother entry to one of the most exciting fields in Computer Science.

Machine learning and Python – a dream team

The goal of machine learning is to teach machines (software) to carry out tasks by providing them a couple of examples (how to do or not do the task). Let's assume that each morning when you turn on your computer, you do the same task of moving e-mails around so that only e-mails belonging to the same topic end up in the same folder. After some time, you might feel bored and think of automating this chore. One way would be to start analyzing your brain and write down all rules your brain processes while you are shuffling your e-mails. However, this will be quite cumbersome and always imperfect. While you will miss some rules, you will over-specify others. A better and more future-proof way would be to automate this process by choosing a set of e-mail meta info and body/folder name pairs and let an algorithm come up with the best rule set. The pairs would be your training data, and the resulting rule set (also called model) could then be applied to future e-mails that we have not yet seen. This is machine learning in its simplest form.

Of course, machine learning (often also referred to as Data Mining or Predictive Analysis) is not a brand new field in itself. Quite the contrary, its success over the recent years can be attributed to the pragmatic way of using rock-solid techniques and insights from other successful fields like statistics. There the purpose is for us humans to get insights into the data, for example, by learning more about the underlying patterns and relationships. As you read more and more about successful applications of machine learning (you have checked out `www.kaggle.com` already, haven't you?), you will see that applied statistics is a common field among machine learning experts.

As you will see later, the process of coming up with a decent ML approach is never a waterfall-like process. Instead, you will see yourself going back and forth in your analysis, trying out different versions of your input data on diverse sets of ML algorithms. It is this explorative nature that lends itself perfectly to Python. Being an interpreted high-level programming language, it seems that Python has been designed exactly for this process of trying out different things. What is more, it does this even fast. Sure, it is slower than C or similar statically typed programming languages. Nevertheless, with the myriad of easy-to-use libraries that are often written in C, you don't have to sacrifice speed for agility.

What the book will teach you (and what it will not)

This book will give you a broad overview of what types of learning algorithms are currently most used in the diverse fields of machine learning, and where to watch out when applying them. From our own experience, however, we know that doing the "cool" stuff, that is, using and tweaking machine learning algorithms such as support vector machines, nearest neighbor search, or ensembles thereof, will only consume a tiny fraction of the overall time of a good machine learning expert. Looking at the following typical workflow, we see that most of the time will be spent in rather mundane tasks:

- Reading in the data and cleaning it
- Exploring and understanding the input data
- Analyzing how best to present the data to the learning algorithm
- Choosing the right model and learning algorithm
- Measuring the performance correctly

When talking about exploring and understanding the input data, we will need a bit of statistics and basic math. However, while doing that, you will see that those topics that seemed to be so dry in your math class can actually be really exciting when you use them to look at interesting data.

The journey starts when you read in the data. When you have to answer questions such as how to handle invalid or missing values, you will see that this is more an art than a precise science. And a very rewarding one, as doing this part right will open your data to more machine learning algorithms and thus increase the likelihood of success.

With the data being ready in your program's data structures, you will want to get a real feeling of what animal you are working with. Do you have enough data to answer your questions? If not, you might want to think about additional ways to get more of it. Do you even have too much data? Then you probably want to think about how best to extract a sample of it.

Often you will not feed the data directly into your machine learning algorithm. Instead you will find that you can refine parts of the data before training. Many times the machine learning algorithm will reward you with increased performance. You will even find that a simple algorithm with refined data generally outperforms a very sophisticated algorithm with raw data. This part of the machine learning workflow is called **feature engineering**, and is most of the time a very exciting and rewarding challenge. You will immediately see the results of being creative and intelligent.

Choosing the right learning algorithm, then, is not simply a shootout of the three or four that are in your toolbox (there will be more you will see). It is more a thoughtful process of weighing different performance and functional requirements. Do you need a fast result and are willing to sacrifice quality? Or would you rather spend more time to get the best possible result? Do you have a clear idea of the future data or should you be a bit more conservative on that side?

Finally, measuring the performance is the part where most mistakes are waiting for the aspiring machine learner. There are easy ones, such as testing your approach with the same data on which you have trained. But there are more difficult ones, when you have imbalanced training data. Again, data is the part that determines whether your undertaking will fail or succeed.

We see that only the fourth point is dealing with the fancy algorithms. Nevertheless, we hope that this book will convince you that the other four tasks are not simply chores, but can be equally exciting. Our hope is that by the end of the book, you will have truly fallen in love with data instead of learning algorithms.

To that end, we will not overwhelm you with the theoretical aspects of the diverse ML algorithms, as there are already excellent books in that area (you will find pointers in the Appendix). Instead, we will try to provide an intuition of the underlying approaches in the individual chapters—just enough for you to get the idea and be able to undertake your first steps. Hence, this book is by no means *the definitive guide* to machine learning. It is more of a starter kit. We hope that it ignites your curiosity enough to keep you eager in trying to learn more and more about this interesting field.

In the rest of this chapter, we will set up and get to know the basic Python libraries NumPy and SciPy and then train our first machine learning using scikit-learn. During that endeavor, we will introduce basic ML concepts that will be used throughout the book. The rest of the chapters will then go into more detail through the five steps described earlier, highlighting different aspects of machine learning in Python using diverse application scenarios.

What to do when you are stuck

We try to convey every idea necessary to reproduce the steps throughout this book. Nevertheless, there will be situations where you are stuck. The reasons might range from simple typos over odd combinations of package versions to problems in understanding.

In this situation, there are many different ways to get help. Most likely, your problem will already be raised and solved in the following excellent Q&A sites:

`http://metaoptimize.com/qa`: This Q&A site is laser-focused on machine learning topics. For almost every question, it contains above average answers from machine learning experts. Even if you don't have any questions, it is a good habit to check it out every now and then and read through some of the answers.

`http://stats.stackexchange.com`: This Q&A site is named Cross Validated, similar to MetaOptimize, but is focused more on statistical problems.

`http://stackoverflow.com`: This Q&A site is much like the previous ones, but with broader focus on general programming topics. It contains, for example, more questions on some of the packages that we will use in this book, such as SciPy or matplotlib.

`#machinelearning` on `https://freenode.net/`: This is the IRC channel focused on machine learning topics. It is a small but very active and helpful community of machine learning experts.

`http://www.TwoToReal.com`: This is the instant Q&A site written by the authors to support you in topics that don't fit in any of the preceding buckets. If you post your question, one of the authors will get an instant message if he is online and be drawn in a chat with you.

As stated in the beginning, this book tries to help you get started quickly on your machine learning journey. Therefore, we highly encourage you to build up your own list of machine learning related blogs and check them out regularly. This is the best way to get to know what works and what doesn't.

The only blog we want to highlight right here (more in the Appendix) is `http://blog.kaggle.com`, the blog of the Kaggle company, which is carrying out machine learning competitions. Typically, they encourage the winners of the competitions to write down how they approached the competition, what strategies did not work, and how they arrived at the winning strategy. Even if you don't read anything else, this is a must.

Getting started

Assuming that you have Python already installed (everything at least as recent as 2.7 should be fine), we need to install NumPy and SciPy for numerical operations, as well as matplotlib for visualization.

Introduction to NumPy, SciPy, and matplotlib

Before we can talk about concrete machine learning algorithms, we have to talk about how best to store the data we will chew through. This is important as the most advanced learning algorithm will not be of any help to us if it will never finish. This may be simply because accessing the data is too slow. Or maybe its representation forces the operating system to swap all day. Add to this that Python is an interpreted language (a highly optimized one, though) that is slow for many numerically heavy algorithms compared to C or FORTRAN. So we might ask why on earth so many scientists and companies are betting their fortune on Python even in highly computation-intensive areas?

The answer is that, in Python, it is very easy to off-load number crunching tasks to the lower layer in the form of C or FORTRAN extensions. And that is exactly what NumPy and SciPy do (`http://scipy.org/Download`). In this tandem, NumPy provides the support of highly optimized multidimensional arrays, which are the basic data structure of most state-of-the-art algorithms. SciPy uses those arrays to provide a set of fast numerical recipes. Finally, matplotlib (`http://matplotlib.org/`) is probably the most convenient and feature-rich library to plot high-quality graphs using Python.

Installing Python

Luckily, for all major operating systems, that is, Windows, Mac, and Linux, there are targeted installers for NumPy, SciPy, and matplotlib. If you are unsure about the installation process, you might want to install Anaconda Python distribution (which you can access at `https://store.continuum.io/cshop/anaconda/`), which is driven by Travis Oliphant, a founding contributor of SciPy. What sets Anaconda apart from other distributions such as Enthought Canopy (which you can download from `https://www.enthought.com/downloads/`) or Python(x,y) (accessible at `http://code.google.com/p/pythonxy/wiki/Downloads`), is that Anaconda is already fully Python 3 compatible—the Python version we will be using throughout the book.

Chewing data efficiently with NumPy and intelligently with SciPy

Let's walk quickly through some basic NumPy examples and then take a look at what SciPy provides on top of it. On the way, we will get our feet wet with plotting using the marvelous Matplotlib package.