Getting started in Machine Learning as a Pure Mathematician

Daniel Platt

1 Nov 2021

Python Programming

If you know no programming at all, www.py4e.com is a great start. It comes with tons of exercises. After this course you must know about: variables, if/for/while, functions and classes. If you have never seen these things, you will need a long time to wrap your head around, probably longer than for all the Machine Learning stuff. It's fine if you need Google to solve the exercises, for example to look up the syntax of a "for" loop. But to know how to Google for it, you must have seen it once.

If you know some programming language, but you don't know Python then reading through the official tutorial and referring back to it during the first months or so when writing your code is probably enough: https://docs.python.org/3/tutorial/

ML basics

1.

I recommend learning about Machine Learning using neural networks first. Once you've understood this, it is easy to look up another machine learning algorithm and apply it. The general procedure (data cleaning/training set/test set) works the same for different algorithms. These two YouTube videos are the best at explaining the (very simple) theory of neural networks:

But what is a neural network? | Chapter 1, Deep learning
https://www.youtube.com/watch?v=aircAruvnKk
Gradient descent, how neural networks learn | Chapter 2, Deep learning
https://www.youtube.com/watch?v=IHZwWFHWa-w&t=747s
The series has two more videos that are less important at first, you can revisit them later.

The more difficult part is the practice of machine learning.

2.

There are many courses. One that I checked myself and is reviewed very positively is Andrew Ng's "Neural Networks and Deep Learning" on Coursera: https://www.coursera.org/learn/neural-networks-deep-learning . (You can sign up for the trial and immediately cancel the trial and keep access for 7 days.) There you create your own neural network from scratch. This is not strictly necessary to do machine learning, as there are programming libraries (such as tensorflow) that do this for you. But you're a mathematician, so you probably want to have some understanding of what's going on under the hood.

3.

Most important is that you know how to use some programming library. The easiest to use is Keras, which is a simplification of tensorflow. The alternative is PyTorch, which is more difficult to use, but unfortunately most commonly used for geometric deep learning. I recommend completing one project in Keras (see below). Switching to other

libraries is not hard then. Start with this one for Keras: https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/ . It's good because it uses no built-in dataset but an external .csv file. In the beginning, the difficulty in machine learning will be loading and processing your data correctly, not coming up with clever machine learning models.

Complete some ML project

1. Use a Neural Network to participate in this competition: https://www.kaggle.com/c/titanic . (A neural network is actually a bad choice for this problem, but it's still good for practice.) You'll have to do some data cleaning here before you can use it for your neural network. For example "male", "female" must be converted into numbers, and you must do something with missing values, for example replace them with 0. This will be the most difficult part here. Usually, people use the Python library "pandas" for this. This has the benefit that you can search things like "pandas make text variables numbers" and get useful results like this: https://pbpython.com/categorical-encoding.html . Kaggle also lets you look at many solutions by others which you probably want to do to help you get started.

2. ML with synthetic data
Consider one of the following problems:
(a) compute the determinant of a 10x10 matrix
(b) check if the multiplicity of the solutions of a quadratic equation az^2+bz+c=0 is 1 or 2 (from section 3.1 in https://arxiv.org/pdf/2101.06317.pdf)
(c) check if a graph contains an Euler circle (from section 3.3 in https://arxiv.org/pdf/2101.06317.pdf)
Generate random data for one of these problems. You'll have to Google how to generate random things in Python. And for computing determinants or checking if a graph contains an Euler circle, use existing libraries if possible. For example: searching "python check euler circle" finds this result: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.euler.eulerian_circuit.html . NetworkX is often used in graph machine learning problems, so it's not wasted time to learn about it, even though it will take a substantial amount of time to learn how to use NetworkX for this task.

3. Learning Hodge Numbers
Reproduce the experiment from section 3.2 in https://arxiv.org/pdf/1706.02714.pdf . The data can be download here: http://www-thphys.physics.ox.ac.uk/projects/CalabiYau/cicylist/ (archived here: https://web.archive.org/web/20211030164405/http://www-thphys.physics.ox.ac.uk/projects/CalabiYau/cicylist/). The problem is to learn a map from 12x15 matrices to the first Hodge number. The most difficult part here will be the data cleaning, during which you must parse the text document to give you the matrices and Hodge numbers.

Software Development

People have strong opinions about what editor to use. I use PyCharm and am happy with it.

Everyone uses the version control system git. Try to get used to it by creating an account on github and using it already for your first experiments.

Certificates/projects

Some unsolicited advice: many online courses offer certificates. Of course, they are nothing compared to a computer science degree. But for a pure mathematician it's a small useful thing to have. Google Cloud's "Professional Data Engineer" https://cloud.google.com/certification/data-engineer is also a mildly useful one to have, I've been told. Amazon's AWS also has some certificates but I can't recommend a particular one. I imagine there are a few managers that think such certificates are silly, but so far everyone I spoke with said they are a small plus for an application.

Apart from that: it's common to put projects on your CV in computer science. If you don't have a computer science publication, you must have such a thing when applying for a machine learning job. Common things are "fun" web applications, like detecting the facial expression of an uploaded photo. Another alternative I can think of is taking an experiment from https://arxiv.org/pdf/2101.06317.pdf and improving the accuracy (by using some more sophisticated neural network) and writing that up and putting it on your website (probably not interesting enough for Arxiv). If you're interested in natural language processing, there are some silly possibilities there, such as sentiment analysis of research articles (cf. https://realpython.com/sentiment-analysis-python/), or automatically generating research article abstracts (cf. https://stackabuse.com/text-generation-with-python-and-tensorflow-keras/). Project need not be serious.

You should also put knowledge of tensorflow on your CV. Some people may wonder why it's not there, if it's not there.

My background: I learned Python programming for one year in high school and Java programming for a year in university. (Major=Mathematics, Minor=Computer Science) From Sep 2017 until Oct 2021 I did a PhD in differential geometry. I started learning about machine learning in 2018. As a first project, I decided to recreate the "Learning Hodge Numbers of Calabi-Yau manifolds" (see below) project in PyTorch together with a friend in Jun 2018. We failed terribly for a while and got the first working version in Keras after some months. At that time my PhD didn't go well, and I was 100% certain I don't want to stay in academia but go into industry, and software/machine learning seemed to be the most promising option except for finance. At the time of writing, I am working as a postdoc in differential geometry (didn't see that coming).