ML Roadmap

By the Brain and Cognitive Society, IITK
bcs.iitk@gmail.com
Discord

So You Want To Get In On The AI Buzz?

Even your grandmother probably has heard about ChatGPT by now, and the world is constantly muttering words like AI, Machine Learning, Transformers, Convnets, Neural Networks, GPT, and god knows what else.

This guide contains an extremely large list of resources, designed (hopefully) not only to help you figure your way out around the ML world but also to help you find out where you might fit into it. If a section feels too daunting, try coming back to it later. If you find an overly interesting section, go deeper into it. We do not include a timeline because you really shouldn't feel the stress of falling behind some notion of an 'average' learner. So feel free to spend an hour a day on this if you think discipline is your game, or maybe finish it over a couple of all-nighters and some nice strong coffee.
This guide tries to cover a large variety of resources at the cost of having a specific learning plan or path. This means you should cherry-pick resources from here and figure out your own learning plan. Try out different things, but stick to whatever you like best. There will likely be considerable overlap between resources, so you should skim or gloss over things whenever you need them. The other advantage is that even if you've already started with your ML journey, you can still find useful things here if you want to get better at some specific topic. Honestly? The absolute worst way to go about this guide would be to check things off one by one.

Quick Guide to the Guide:

1. Every section has a little writeup about what the goals of the section are along with a list of resources of various types - books, online courses, YouTube videos, (reddit threads?). You need not go through all the resources. Use the type of resources you like best. Someone had to research a lot to make sure you get the benefit of this choice, don't waste it.

2. The Words List is something you should read AFTER you're done with the resources of a section. You should at the very least recognise all the words in the word list, and be able to Google out relevant documentation within a search or two. It is essentially a very basic form of self-assessment. If you do not recognise a word or two from the list, don't panic. You can just use Google to read up about it. If this happens for most of the words though, then maybe you didn't pay enough attention or need to read more resources. Oh also they are in no particular order.

3. An asterisk on a resource(*) means the resource is not free. I've tried to stick to free resources as much as possible, but sometimes there's a paid resource that is worth mentioning. Of course, if you're familiar with r/Piracy, then that asterisk probably doesn't mean much to you does it?

4. Stay away from ChatGPT while you are still learning things. ChatGPT, Copilot and other AI tools will be your best lieutenant in the real world, but if you try to learn alongside it, you will either feel like a fraud or a god (depending on your self-confidence) and neither is great for learning. Google a lot. Read a lot. If you don't understand something, ask how to understand it better. But do not use ChatGPT to gloss over it. There'll be plenty of time for you to use LLMs on the job, and the best part of ChatGPT is that unlike googling for things, the learning curve to get the most out of it is tiny.

5. This is not the best guide for ML out there. There doesn't exist such a thing. How you learn best is something rather unique to you, and to tailor an experience

specifically for you is frankly... quite hard. If you don't like the way a resource is, ask for alternatives. One of the good things about the booming popularity of ML is that you can always get help. Mix and match, swap out things you don't like, swap in things you do.

Section 1: Python

While there exist multiple languages that are used by Machine Learning Engineers, the best one to start with, and the most widely used one is without a doubt, Python. Which is great if you've never coded before, because Python is extremely easy as a first language. The syntax is extremely simple, and you will mostly just feel like you're explaining what you want in English.

You do not need to become a Python professional before you can start with ML though. Understand the basics, how to use the standard data structures like lists, tuples and dictionaries, how to write a function, maybe the very basics of objects and classes. When you feel like you have a handle on the language, feel free to go to the next section.

We know. It is not the fastest language, but frankly, people far smarter than us have already written down Python libraries in C to handle anything that needs speed. If you feel like you can do it faster than them, then you can always learn C later.

Watch Out For!

- We mean Python 3. Python 2 is deprecated and you should NOT be writing any new code in Python 2. Also Python is offered by various distributions, and I recommend Anaconda, since it's one specifically themed for data science, and has great environment support.
- If you are coming over from a different language, try not to blindly translate the way you code in that language to Python, instead put some effort into

understanding how to write *Pythonic* code. This will make things much more fun and much simpler.

- You don't *need* an IDE, but it might be good to set one up properly. I recommend VS Code, but there's tons on offer. More importantly perhaps, you should get used to Google Colab. It's a cloud based environment (so it runs on a potato PC) where the majority of ML and DS related work happens.
- There are two main mediums of working with Python code - one is the standard script files (.py), the other are Python notebooks (.ipnyb). Scripts are better when you're just starting with Python, but which form factor you use more often depends heavily on what exactly you're doing in the ML community. For now, try to be sort of comfortable with both environments.
- The resources on NumPy, Pandas and Matplotlib can be done later. You'll run into these libraries in Section 3. But they're quite fun to play with, so if you're falling in love with Python, then maybe it's worth figuring out how to get xkcd mode working in matplotlib as well?

Resources

- If you don't know programming -
  - Python Like You Mean It - Ryan Solanski, aimed specifically for STEM applications[EBook]
  - Automate the Boring Stuff - [EBook]
  - The Hitchhiker's Guide to Python - A very Pythonic-python practice handbook [Interactive Book]
  - Think Python - Allen B. Downey on O'Reily [Book*]
- If you know programming
  - The Official Python Tutorial - [Docs]
  - Learn Python in Y Minutes - [Digital Book]
  - Tiny Python Projects - Tiny Python Projects. Duh. [Tutorial]
  - Ultimate Python Study Guide - [Github]
  - Codewars - Coding challenges across various levels of difficulty [Website]
- Tools that might help everyone

    - Python Tutor - Lets you visualize the step by step execution of code
    - r/learnpython - The Reddit for advice on learning python
    - Beyond PEP-8 - Raymond Hettinger, Best practices for beautiful intelligible code
- NumPy, Pandas, and Matplotlib -
    - The official quick start guides for NumPy, Pandas and Matplotlib [Docs]
    - Python Data Science Handbook - The basic handbook on Data Science with Python. [E-book]
    - 101 NumPy exercises for Data Analysis [Tutorial]
    - Pandas from the Ground Up - Brandon Rhodes at PyCon 2015 [Youtube]

Word List

File modes, filter(), Generators, Immutability, List Comprehension, map(), Variable Scope, Vectorization, cmap, np.linspace(), reshape(), merge(), Data Munging, Broadcasting, Truth Values of Non-Boolean Objects, Lambda Functions, zip(), any(), __main__,

Section 2: Math

Machine Learning is basically math. Sure it may seem (big hefty emphasis on seem, the deeper you go, the more you realize this entire field is just Linear Algebra and Probability on steroids) unnecessary, but without math, you will never really be able to build any intuition about what is going on. And without intuition, you won't really be designing any models, even if you may be able to follow my instructions. Even if you would like to follow a top-down approach to learning ML, I'd still recommend doing this section before the last one. Because a solid understanding of the math involved can be the difference between the algorithms feeling pointlessly vague or feeling trivially simple. What fun is learning ML if you can't at the very least try to explain some part of its black box?

Thankfully though, the average ML Engineer needs pretty basic math. You need to understand multivariable calculus, linear algebra and some probability and statistics. There's really 3 parts to this as I see it, there's the theory, the intuition, and the application to ML.

Data in Machine Learning contexts is frequently represented using n-dimensional arrays, and Linear Algebra helps you understand what mathematics looks like when playing around with such things. Multivariable calculus comes in when you have functions on these arrays and need to figure out a way to optimize them. Finally probability theory comes in to explain why these ML algorithms work, and what exactly is being optimized. Hopefully after you're done with these resources, you'll be able to better phrase the importance of each of these domains.

For the most part, if you truly paid attention during the first year math courses at IITK, you can assume you have basic math knowledge (in terms of theory) and use a resource like the Mathematics for Machine Learning book. But if you didn't (honestly very understandable), go easy on yourself and read a book or do a course on the topics to understand the basics before moving on to the applications if you don't want to handwave every single thing.

(For readers in the Mathematics and Statistics Department, your second year courses will usually cover each domain in really solid detail. Feel free to rely on those for theory.)

Resources

- 3Blue1Brown - Arguably one of the best math explainers out there on YouTube. Definitely the best way to get a solid intuition on anything math. He already has playlists on each of these domains, and on Neural Networks itself [Youtube Channel]
- MIT OpenCourseWare - If MIT is doing it, surely there must be something right. Since course offerings are updated, we are not linking a particular course. But these versions are considered quite good at the time of writing this -

    - Linear Algebra Fall 2011 - Taken by Gilbert Strang himself. His book on the subject is considered one of the Bibles. [Course]
    - Probabilistic Systems Analysis And Applied Probability - A bit harder than necessary, especially if you don't have a math background, but extremely rigorous and comprehensive as a result. [Course]
    - Matrix Methods - Also taken by Gilbert Strang, and strings the three branches together to make a course more designed for its specific applications in ML [Course]
- DeepLearning.AI's Mathematics for ML - DeepLearning.AI will come up later as well, considering it basically gives us the bible for deep learning and neural networks, but their math offering is pretty solid too. [Online Course]
- Jon Krohn's Youtube Channel - Weightlifter who teaches math. Surely that's enough to convince you. He has playlists on all three Linear Algebra, Calculus and Probability. I think he's done a particularly fabulous job with the Calculus [Youtube]
- Mathematics for Machine Learning - Good if you feel like you already have foundational math knowledge in these areas. It tries to stitch together these three into ML and does a great job at it. Might be too hard as a first read though. On the other hand if you have really strong fundamentals in math, you might as well ignore a book like this since ML courses will also cover a lot of this stuff. [Book]
- Elements of Statistical Learning - Often comes up as a sister to the previous resource. If you don't like one you'll like the other sort of a thing. It still assumes a lot of foundational math knowledge though. Also believe this one's better if you're specifically going to focus more on the Data Science elements (see section 4) [Book]

If you fell in love in this section, then good news for you, you could get into extremely pure-ML research. Like the kind of people who design new architectures and stuff. Solving questions like how to make *inherently* better and faster models, regardless of the actual data. If you want to go forward in this direction, you should dive deeper into the math, and possibly also on how things are implemented in low level languages. Try resources like Computational Linear Algebra [Youtube Playlist]

Word List

Basis of a Vector Space, Eigenvalues and Eigenvectors, Partial and Total Derivatives, Lagrange Multipliers, Gaussian Distribution, Prior and Posterior Probability, Orthonormalization, Poisson's Distribution, Correlation, p-value, Confidence Intervals, Lagrange Interpolation, Central Limit Theorem

Foundational Machine Learning

It's time to actually get to the ML bit. To make things easy, there is universal agreement that Andrew Ng has the single best foundational ML offering. Pick it with your eyes closed.

Most of you who have started off on this journey will probably do the CS771 course offered at IITK. It is a perfectly great course, but by no means is it a substitution for this section. It has a more theoretical nature for the most part but does not cover quite enough for it to be considered comprehensive. Do I still recommend that you take Purushottam Kar's offering though? Absolutely. He's a pretty brilliant man, and seeing his excitement for the subject, along with his insights in specific problems has been one of my best experiences with a course in IITK.

Usually guides will try to split this section into more specific sub-sections - Linear and Logistic Regression, SVMs, CNNs, etc. Most resources in this section will already do this internally, so I am not breaking it up in the guide itself.

Resources

- Andrew Ng's Courses - If there's a gold standard for this section, it has to be this one. ML nerds will usually call his courses the 'Holy Trinity of ML'. Keeps things rigorous but never makes it too mathematical either.
    - Machine Learning Specialization - Part One of the Trinity [Course]
    - Deep Learning Specialization - Part Two of the Trinity [Course]

    - Tensorflow in Practice - Part Three of the Trinity [Course]
    - Stanford CS229: ML - His lectures at Stanford, much more mathematical than the above courses and hence only meant for you if you want to really get into theoretical ML. [Youtube Playlist]
- Yoshua Benigo's Deep Learning Bible - The other extremely common resource which is thrown out on every reddit. It is quite mathematical and a bit dated in my opinion. But if you say you've read it properly you'll get quite a bit of instant respect. [Book]
- Dive into Deep Learning - E-book that is surprisingly comprehensive and up-to-date with respect to code implementations (PyTorch, Tensorflow and JAX as well). I don't believe it is the best when it comes to intuition, but it is great if you're coming back to give ML a second shot and have a sort of hazy knowledge about things. [E-book]
- fast.ai - This one is also a really good resource, but in my opinion its fatal flaw is that it teaches the subject based on its own coding framework, which is kind of non-standard. Does do a great job of teaching things in the top-down way though. [Online Course]
- MathematicalMonk - I was not exactly sure about whether to put this in the math section or this one, but I think it is more of a foundational ML playlist, since it explains ML concepts and the mathematics of how it works. It's got that Khan academy style vibe. Again I only recommend this for those more mathematically inclined. [Youtube Playlist]
- Neural Network Playground - This isn't a resource in itself, it is more of a companion tool. Effectively this lets you tweak the parameters of a basic neural network and see the effect it has on your model's predictions on some nice data. If I could learn ML from scratch again, this is where I would build the intuition for it.

If you're interested in something even more mathematical, something to help you design new ML algorithms, you might want to check out Deep Learning Architectures by Ovidiu Calin. There's also Geometric Deep Learning, which hopes to be the next big thing, even if our hardware isn't quite good enough yet.

Word List

Logistic Regression, SVM, One-hot encoding, Regularization, Norms, Decision Trees, RNNs, kNNs, PCA, Confusion matrix, Cross-validation, Bias, Clustering, Leaky ReLU, Gaussian Kernel, MLE, Decision Boundary

Section 4: Practice

This isn't a section to do all at once, rather I'd encourage you to interleave it once you've started to get a handle on Section 3.

There's so many datasets available online, you just have to put in the effort to find it sometimes. If you don't give a shit about house prices, then don't. Find something better to regress on.

This is also the part of the guide where you actually learn to do solid Exploratory Data Analysis. EDA is simply gathering insights about your data so as to help you decide things like what variables are relevant, what kind of ML models might work well on this data, if some variables can be changed into more useful ones, etc. It's something best learned through practice. Your first couple of practice problems will probably have you feeling lost about what to do. You'll probably then want to see what other people did with the data, and see how they make inferences using data analysis to decide how they want to solve the problem. And then slowly you'll learn to recognise the patterns in new datasets and gain confidence on what to do when you see a problem without needing the help of others. If you don't understand something, ask! Or maybe just fuck around in a copy of the code and find out what's going on. This is how I personally learned EDA.

But if you want something more concrete, I've linked up some resources specifically for EDA in this section too.

A note on MNIST: If you decide to google datasets to practice on, you're likely to run into MNIST at some point. Particularly, MNIST Handwritten Digits. These are rather old and small datasets, and I don't think they are still worth doing in the modern day. The direction of AI growth has always been towards larger and larger datasets, and working with MNIST datasets will give you a false sense of understanding, since these are very easy to get amazing accuracy on with hardly any tweaking.

A note on Kaggle: Kaggle is a learning tool. It is not a project. Think of Kaggle more like an ML simulator. It's a cleaned out experience best for learning, but in the real world there's other things you'll have to learn, like scraping and cleaning data, accuracy vs computational cost tradeoffs, constraints on every part of your pipeline and so much more. I highly recommend not relying on Kaggle alone, especially after you're done with Section 3 and instead doing a more serious project from scratch. Without the idea pre-planted in your head in the form of a contest and without the wealth of submissions available to you.

Resources

- Kaggle - You will likely need to use Kaggle at some point of time in your ML journey. It's kind of like an analogue to codeforces but for ML. There's a solid array of contests and a very active community, including those Chinese dudes who are going to be better than you at everything so you can cry about your life. The unfortunate downside of kaggle is that because it is so popular, doing a project here feels more like processing a mass-manufactured product. These are some contests that are either extremely common or are quite liked by me.
    - Titanic - Where everybody begins. The first contest that almost everybody who knows ML gave, if they gave one. [Contest]
    - House Prices - Another extremely common contest. Honestly bored of seeing this everywhere though. [Contest]
    - Taxi Fare Prediction - This has also been around a long long time, although I don't see it being talked about nearly as much. [Contest]
    - San Francisco Crime Classification - I love this one. The amount of things you can do with this dataset is brilliant. The domain is slightly unfamiliar, but relatively easy to get insights on. [Contest]
    - Gene Expression Dataset - Something a little biological, since people don't appreciate the biological applications of ML nearly enough. You

can use it to classify patients with leukemia, or maybe something else entirely? [Dataset]
    - Women's Shoe Prices and Men's Shoe Prices- Feel like brand X is totally overpriced? Or maybe you want to prove the existence of the pink tax? These datasets were made for you.
- Google Datasets - Google has made a bunch of datasets available to the public. Its popularity is on the rise, and the quality of datasets is quite good. Totally worth trying to find a good dataset to work with here.
- Data is Plural - A newsletter that curates a lot of really interesting datasets.
- r/datasets - Reddit never lets you down huh.
- Exploratory Data Analysis Resources -
    - Exploratory Data Analysis - John Tukey is considered the father of EDA, so this is one of the OG greats when it comes to this subject [Book*]
    - Think Stats - Allen Downey is usually the other recommendation that the Internet agrees on. It's like a Primer for Probability and Statistics in ML. [Book]
    - The Leaderboard and Discussion Tabs in Kaggle - There's always a wealth of information here if you can sort through it.

Word List

Virtual Environments, gdown, GPUs and TPUs, EDA, Heatmaps, Boxplots, Violin plots
Cross-Entropy Loss, NLP, CV, RL, Gradient Boosted Decision Trees, Imputation, tSNE

The End. Almost.

You are at the end of this guide. If you've made it this far, you're ready to try your hand out at more serious projects, so it's time to search for project ideas. Or maybe you want to become the Kaggle master and absolutely wreck contests. I hope you also got a better idea of where in the ML pipeline you think you might fit in, whether that's designing architectures, ML ops, DS, NLP, Vision, RL, BioML or whatever. In any case, we wish you the very best of luck.

In case you hated this roadmap, here are some others -

- P Club's ML Roadmap - Made with love and memes by our sister club
- ZuzooVn's Roadmap - A Github repo that deals with the roadmap question in a different style
- An Interactive, Visual ML Roadmap - if you want a roadmap organized in a flowchart.

Happy Learning!

Any doubts? Shoot 'em up on discord

- Debarpita Dash(dd1904)
- Manasvi Nidugala(manasvi_nid)
- Sagar Arora(qu.bit)
- Udbhav Agarwal(Udbhav_agarwal)