Software engineering for data science

Course Description

Software engineering in data science requires a wide variety of technical skills. Not only ones need to know how to build a model for their data, but also engineer around the model from continous ingestion of new data to produce a deployable product. This course aims to expose the students to many modern software engineering practices for software engineering. We will start with problem analysis, programming languages, toolings such as version control, testing, debugging, to MLOps such as data base interaction and experiment tracking, all the way to open source developement cycle and governance, continous integration and deployment, and user interface and experience design. After this course, the student should have working knowledge to design and start an end-to-end data science project. This course focuses on hands-on experience, labs and tutorials, which the students will design and develop a semester-long project with the guidance of the instructor. Prerequisite(s): prior experience with programming is highly recommended.

While each students should have their own project for the course, students are encouraged to work with each other. The most important part of this course is to have fun, and by the end of the course the students should have build some small products that will be available to the general public.

Philosophy

I choose to teach everything but building neural network

If you want to learn how to build a certain neural network, this is not the course, I have another course that focuses on deep learning engineering. This course is about everything around neural network, which in my opinion is way harder to get into and definite need a mentor more than building neural network. Most of the neural network library such as PyTorch and Jax are very well-documented, and there are tons of great blog post teaching you how to do it. The workflow of building a neural network is usually quite well defined, get data, tune numbers, publish paper. There are not much choices you have to make, and for the choices you do have to make, often you can rely on your mathematical intuition to guide yourself. On the other hand, mathematical insight is not going to be immediately useful when we are talking about how to setup a CI/CD pipeline, or how to write a good documentation. Navigating the design space of neural network is relatively straight forward, but navigating the design space of a full fledge software is like kayaking in a stormy sea. I believe taking this course first before you dive into deep learning specific engineering will benefit you tremendously, since you will be equipped with the tools that bring your neural networks to the world and make a difference.

I am just your guide

There are different schools of thought when it comes to building software, just look at how many javascript frameworks and linux distro there. In the end, it is more about what you are trying to achieve and your style more than anything. There are general principles, best practice and anti-pattern, but a lot of engineering is about trade-offs and taste. I am structuring this course and the material according to my taste and preference, which is aligned with scientific computing and open source software developement. This may not be the best option in a different environment, such as in cyber security space. I am designing to course not only to show you the tools and workflows I prefer, but also why I pick the way I program. Style evolves over time as well, so remember to be agile and be ready to change if betters options are available.

Software engineering for data science

I can help you to get your A, but you have to get it yourself

Since this is a class that suppose to teach you practical skills instead of the truth of the universe, I do not see the point in quizing and grading you. Instead, I value whether you can build some software and serve the community in the end, so that is how the assessment scheme is going to work: for the official grading, we will set up some simple labs for each topics, as long as you have done all of them, you will get an A. However, the true A in this class will be a software that you build and make it available to the world, and I will not be the (only) judge of whether you get an A or not. Alongside with the labs, you will pick a software project at the beginning of the course and build it as we progress in the class. At the end of the course, we will have a showcase of everyone's project. This **do not** contribute to your grade in any way. The catch is, hopefully you pick something that you are truly passionate about instead of just something that you think will be easy to build for the class, and you can actually show the world what software you have made. As a continuation of the course, I will help you publish and leverage your software for your career path, whether that involves publishing a code paper, or submitting to conferences. In the end, I think the grading is mutual, the grade for my teaching is how much fun you have, and the grade for your coursework is how proud of your software you are.

Use of AI-assistant

I am very okay with students using AI-assistant tools such as copilot or ChatGPT in labs and projects. I uses copilot extensively in my daily workflow, and it has been a great productivity boost to what I do. In the end, your clients will not care whether you use copilot or voodoo magic to cook up the products, all they care is whether the product is good or not. On top of that, as much as I want AI to take over my job, it still kind of fails miserably. I am not even talking about the science bits of it, which we can go into hours of why AI may struggle on that. I am talking about the engineering part of AI. I dropped my ChatGPT subscription quick a while ago, since it just constantly fail to give me an answer that will work out of the box. Instead, I use ChatGPT like a search engine, usually in order to find the keyword that is related to what I want to search, then I just punch them into Google.

My bottom line with AI-assistant is it has to make you a better engineer, not worse. If ChatGPT helps you find new concepts and write codes that are more efficient than what you would have written on your own, by all mean go for it. However, if you start to rely too much on ChatGPT, and produce inefficient code or buggy code, that is an issue. The goal of this course is to make you a capable software engineer in the era which AI-assistant exists, and whatever we can do to achieve that is appericated.

Logistics

Labs

The labs serve two purposes: first, it is good to have some very well defined tasks you can practice whatever we have learning during the lectures. It is beneficial to you to get that muscle memory, at the same time to have some reference points if you want to do the same task in the future. Second, the labs are going to be 100% of your grade. I fundamentally do not value the labs as much as all the other non-graded activites, so my policy is as long as you complete all the labs as intended in time, then you will get an A. They should be relatively straight forward and should not take too much of your time on average. However, let me be clear here that the labs taking up a 100% of the grade does not mean you can slack off the other activites. I make the labs simple in order to make room for all the other fun stuffs, so they are the highlights. If you just want an easy A from doing the labs and planning to go easy on the other activities, I will strongly advise against taking this course.

"Pitch" session and Mid-semester update

At the beginning of the course, you will pick a project that you will work on throughout the semester, which I have a couple examples of what I think is reasonable for this course in terms of size. We will have a pitch session, which everyone will introduce their project to the class, specify the scope and layout a couple of milestones. It is a pitch session in qoutation because there is no seed money for this, but it is a good practice to go through the thought process which helps you laying down a solid plan for the project. In the middle of the semester, we will have an update session which everyone presents their progress they have been makeing to the class. These will not contribute toward your final grade, but I value this more than your lab.

Showcase

By the end of the semester, I think it will be really fun to showcase your tools to each other and people outside the class. I will organize a showcase event which everyone will have an opportunity to present their project. In the ideal scenario, everyone should have some codes together with a demo web app we can all play with. The idea of having the pitch session and the showcase session is to give you the full experience of how to present your projects. Once again this will not count toward your final grade, but I also value this more than your lab.

Attendance

Since the course is very hands-on and we meet only once a week, it does not really make sense to skip the class. I understand life happens from time to time, and if there is very legitimate reason you cannot make it to a particular class, there is a quota for missing the class once, not including the mid-semester update session and the show case session. If you miss the class more than once, every missed class will limit your highest grade by one letter grade, e.g. if you missed the class twice, the highest grade you can get is a B. If you miss the mid-semester update session or the show case session without an extremely legitimate reason, you will not be able to pass the course. I believe coming to the class on time and participate in the class is critical to learning from this course, so I will be completely strict with this policy.

Continuous feedback

Even though I have a particular plan for this course, I think it makes the most sense if you tell me what could be useful to you. I cannot promise I will always be able to accomodate your need by switching up my lecture plan, but I do leave some flexibility when I am designing the syllabus for this course, so there is some chance I can touch on some popular topics. If there is really popular demand, I may also hold additional hacking session outside this course. For more granular feedbacks such as finding bugs in assignments or have general questions, feel free to open issues or pull requests on Github.

Debugging help

As much as I want to promise you everything will be smooth, I would lying if I told you there will not be any bugs and problems during your lab. And during the class, I will probably need to resolve multiple situations in a relatively short time scale. There will be an office hour from the TA the day after the class, which the TA will help you tackle the problems you have during the lab.

Schedule

While there are a few sessions that are going to be a bit different in format, such as the pitch session and the showcase session, the majority of the sessions is going to share the following structure:

Software engineering for data science

1. **Overview**: A brief lecture about the topics of the day (~30 mins)
2. **Lab**: Pre-defined assingments (~30 mins)
3. **Break**: Stretch, get coffee, walk around. Long sitting is unhealthy. (~10 mins)
4. **Hacking**: Integrating newly learned techniques into semester long project (~1 hrs)
5. **Wrap-up**: Summarize and report the progress of the day (~20mins)

During the lab session and integration session, I will go around help people with their problems.

Week 1 **Introduction to the course:** There will not be lab, instead, we will go through the class logistics, and have a brainstorming session help you formalize your semester long project.

Week 2 **Version control:** We will learn how to do version control with git. Since this is at the beginning of the course, instead of a hacking session, we will go around and pitch our project to each other.

Week 3 **Python**: We will introduce python, some best practices in python, and how to set up a package in python.

Week 4 **Julia**: We will introduce julia, and learn about how to set up a project in julia.

Week 5 **Rust**: We will introduce rust, some unique features of rust, and how to set up a project in rust.

Week 6 **Machine Learning frameworks in the three languages**: We will play with three machine larning (-related) libraries in the three languages we learned, including `jax` in `python`, `flux` in `julia`, and `burn` in `rust`.

Week 7 **Mid-semester update:** There will be no lab, and hacking. Instead, people should present their progress so far.

Week 8 **Fall break**

Week 9 **Continous integration:** You will learn how to use actions to continously update and test your code.

Week 10 **Environment management and Containerization:** You will learn how to manage your environment with nix and containerize your applications with docker.

Week 11 **Frontend:** We will learn about the basic of building frontend with svelte.

Week 12 **Database:** You will learn the basic of database and storage.

Week 13 **Open source practice:** You will learn some of the open source practice such as governance and code review.

Week 14 **Fall Recess**

Week 15 **Show case**: There will be no lab and hacking. Instead, we will celerate everyone's hardwork!