Designing & Implementing Data Pipelines for Scientific Research

Lecture 1:
Introduction to Data Pipelines

Dr Ahmad Abu-Khazneh
Senior Machine Learning Engineer
Accelerate Programme
Spring School May 2023

ACCELERATE PROGRAMME FOR SCIENTIFIC DISCOVERY

SCHMIDT FUTURES

About myself & this course

My background

- PhD in Mathematics: applying ML techniques to graph theory conjectures
- Financial Times: implementing NLP data mining to financial reports
- Royal Mail: optimising logistic networks using ML
- Imperial College London: Senior Fellow in Data Science - collaborating with scientists in Faculty of Natural Science use ML in their research, also creating the ML module for MSc in Data Science
- Common thread in my background: pipelines!

About myself & this course

This course

- This is an idiosyncratic course heavily based on my industrial and academic experience
- The focus is on understudied rarely mentioned topics in many ML courses. I observed scientists struggle a lot when using ML in their research due to these blindspots.
- Designing and implementing data pipelines for scientific research is difficult to cover as there is no underlying theory. It is more of a craft than a simple algorithm you can follow.
- It consists of lots of best-practice software engineering principles and is best learnt by guided supervision: exactly what we will be doing in this course!

About myself & this course

Common themes

- Even though this course might strike you as an assortment of topics vaguely related to each other, there is a number of common themes running throughout:
    - Data-centric vs model-centric machine learning
    - Software engineering in small research teams vs large research teams
    - Engineering pipelines in tech companies vs engineering them in academia
    - Most importantly: data pipelines as a means to an end vs the end in itself.

What is a data pipeline

Industrial definition

- No standard definition, each ML practitioner will give you their own definition.
- Let's look at some of the definitions that I have curated, starting with the most common industrial/commercial definition
- "Data pipelines are ETL pipelines"
- ETL stands for Extract-Transform-Load.
- One of the industrial acronyms that does actually make sense as it is perfectly descriptive of a lot of what pipelines do.
- Question for academics: why is 'extract' and 'load' considered as important as 'transform' so as to be put on the same level in the acronym, they seem to be much simpler operations?

What is a data pipeline

Data science definition

- Closely aligned with the ETL definition is the data science definition.
- This is the definition that is implicitly conveyed in many introductory data science courses/bootcamps even if not explicitly stated
- The definition roughly states that the data pipeline is the part of your code in your model that does all the ‘data preprocessing’, for data scientists this mostly involves data cleaning and feature engineering.
- One limitation of this definition is that it is too tied-up to the modelling part - the pipeline is not really seen as an independent entity

What is a data pipeline

Academic definition

- Academic definition: *a data pipeline is the script that I run to obtain the research data in the format that I use for the rest of my analysis* :)
- *Moreover, the script has originally been written by my supervisor but has since been maintained (and mutated!) by multiple generations of her PhD students.*
- *It is quite complex, written in an ancient language, most of us don't understand it 100%... but it does the job.*
- This definition is quite imprecise of course but perfectly relatable for most scientists in academia.
- It captures something of the essence of why pipelines are so important in research but also highlights the tendency of seeing it as a black-box, write once and then slowly increment (or bloat) it over many years from one generation of PhD students to the next.
- If you agree with this definition then one of my motivations for this course is to expand your horizon on what pipelines are and more importantly on *what they can be*.

What is a data pipeline

Programming definition

- Python definition: A pipeline is an sklearn.pipeline.Pipeline object it applies a list of transformers and an optional final estimators.
- This is the most technical definition, I added it here to give you a very concrete example of what it means to be a pipeline.
- This is by no means the only programming object that attempts to capture the concept of pipelines, There are entire programming frameworks that attempt this
- In fact multiple libraries do this in Python alone and some follow a quite different approach from the sklearn approach.
- We will look in more details of the sklearn pipeline later, as it exemplifies multiple good software engineering practice in designing pipelines

What is a data pipeline

Course definition

- As you can see there are a number of different viewpoints on what a pipeline is, and I could have given you many more other descriptions.
- However, I would like to give you my definition which I have developed for this course and similar ones I have given before, and overlaps with many of the previous definitions.

  A data pipeline is a software artefact that consists of all the steps related to preparing data for a scientific study, it is published with its accompanying testing framework, documentation and can easily be installed, forked, extended and deployed.

- The rest of the course will elaborate on each keyword in the definition, illustrating them with examples and outlining some of the related best-practice software engineering techniques for each.

Software artefacts

Software as code vs software as artefact

- The first keyword in the definition is software ‘artefact’. This another common term turns out to be a difficult word to define but it's important to get an idea of what it means as it underlines one of the important themes of this course.
- In particular it speaks to the tendency to view a pipeline as a means to an end vs an end in itself in academia, but a software artefact is something useful beyond its original use for which it was created.
- Hopefully by the end of this course you would have also understood what it means after seeing many examples!
- However, to help academics who are not software engineers gain some intuition to what this means I have developed the following analogy.

Software artefacts

Lecture notes vs textbook

- The analogy concerns something academics are very familiar about: lecture notes vs textbooks.
- To explore this analogy first note that a lot of academic textbooks start life as a set of lecture notes which are created by a lecturer for the specific aim of delivering a particular lecture they have been assigned to.
- In the first instance the notes are implemented by the lecturer in a very ad hoc way - sometimes even hand written and then copied on the board for the class. Eventually a student volunteers to latex (scribe) them.
- The latexed version of the notes help the lecturer make the notes more and more elaborate and polished with each passing year.
- As the lecturer gets more feedback (and time!) she starts fixing typos, inserting more explanations and adding more examples/exercises.
- Eventually the quality of the notes become so high that they start to get circulated in the community and other lecturers adopt them for their own teaching. In particular, ex-students of the lecturer start to use them in their teaching and possibly even enhance them.

Software artefacts

Lecture notes vs textbook

- At some point a publisher or due to the lecturer's own initiative decides to publish the lecture notes into a textbook to make it even more widely available.
- However to become a textbook means doing the following: restructuring the notes into chapters with minimal dependency between them to allow different pathways through the textbook.
- An index is added to make the book useful as a reference for researchers, and solutions are added to make it useful for self-study.
- The important point to note is that their is a lot of effort involved into turning the notes into a textbook even though that effort doesn't substantially increase the core technical content of the original notes,
- However, the changes enable the content to be far more reusable to a much wider and diverse audience. Also the mere fact that it is published is so available from libraries/book shops.
- Thus pipelines as artefacts are the equivalent of turning the messy ad-hoc incremental code that you implemented for your very specific research objectives into something that can be adopted by a much broader audience, in the same way that textbooks undergo a journey from lecture notes to published book.

Why should scientists care about data pipelines

- Well-documented pipelines makes it easy for peer-reviewers to inspect scientist's assumptions and biases (visibility)
- Shareable data pipelines makes it easy for other scientist to run the pipeline on other data (reproducibility)
- Well-engineered data pipelines make it easier for other scientists to extend and build on scientist's work (extensibility / interoperability)
- Well-tested data pipelines increases assurance and satisfies compliancy and privacy controls.
- Some data pipeline can end up more impactful than the actual result they were developed to generate.

Why should scientists care about data pipelines

- Well-maintained data pipelines help deal with *concept drift* and *data drift* which are occur frequently in any research project.
- Lots of journals now require data handling code to pass certain checklists prior to publication.
- Implementing and publishing well-designed, easy to reuse pipelines that can be adopted by a wide variety of scientific domains and scientists increase the chances that your code can be used in the industry.
- So caring about your pipelines is an interesting potential channel for commercialising your research or landing consulting opportunity that can expand your research horizon with industrial partnerships

Intro lab

Example of well-designed pipelines

- In the labs we will be going through some well-designed published pipelines in various domains and provide an overview of their well-designed feature.
- In particular we will have a look at the James Webb Space Telescope (JWST) astronomy pipeline and some of the bioinformatics pipeline in nf-core curated set of pipelines.
- After that, we will go through your own pipelines and highlight some of their limitations and weakness and how they can be improved.