BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Model Deployment

Pravin Y Pawar

Adapted from “Designing Machine Learning Systems” and “Introducing MLOps”

Model Deployment

- “Deploy” is a loose term that generally means making model running and accessible

- To be deployed, model will have to leave the development environment
  - During model development, model usually runs in a development environment
  - can be deployed to a staging environment for testing
  - can be deployed to a production environment to be used by end users

- Production is a broad spectrum
  - For some teams, production means generating nice plots in notebooks to show to the business team
  - For other teams, production means keeping your models up and running for millions of users a day

- If your work is in the first scenario, production environment is similar to the development environment
- If your work is closer to the second scenario, welcome to word of model deployment!

Deployment is easy!

If you ignore all hard parts

- If want to deploy a model for friends to play with, all needs to be done is to
  - wrap predict function in a POST request endpoint using Flask or FastAPI,
  - put the dependencies this predict function needs to run in a container
  - push model and its associated container to a cloud service like AWS or GCP to expose the endpoint

- Can use this exposed endpoint for downstream applications:
  - e.g., when an application receives a prediction request from a user,
  - this request is sent to the exposed endpoint, which returns a prediction

- If familiar with the necessary tools, can have a functional deployment in an hour!

Machine Learning Deployment Myths

- Deploying an ML model can be very different from deploying a traditional software program

- This difference might cause people who have never deployed a model before
  - to either dread the process
  - or underestimate how much time and effort it will take

- Common myths about the deployment process
  - Myth 1: You Only Deploy One or Two ML Models at a Time
  - Myth 2: If We Don’t Do Anything, Model Performance Remains the Same
  - Myth 3: You Won’t Need to Update Your Models as Much
  - Myth 4: Most ML Engineers Don’t Need to Worry About Scale

Deployment : Reality

- The hard parts include
  - making model available to millions of users
  - with a latency of milliseconds and 99% uptime,
  - setting up the infrastructure
    - so that the right person can be immediately notified when something goes wrong,
    - figuring out what went wrong
    - seamlessly deploying the updates to fix what's wrong

- In many companies,
  - the responsibility of deploying models falls into the hands of the same people who developed those models
- In many other companies, once
  - a model is ready to be deployed, it will be exported and handed off to another team to deploy it
- However, this separation of responsibilities can cause high overhead communications across teams and make it slow to update model.
  - It also can make it hard to debug should something go wrong.

Productionalization and Deployment

- A key component of MLOps
  - presents an entirely different set of technical challenges than developing the model
  - domain of the software engineer and the DevOps team

- Organizational challenges in managing the information exchange between the data scientists and these teams must not be underestimated!
  - without effective collaboration between the teams, delays or failures to deploy are inevitable

Model Deployment Types and Contents

what exactly is going into production, and what does a model consist of?

- Commonly two types of model deployment:
- Model-as-a-service, or live-scoring model
  - Typically the model is deployed into a simple framework to provide a REST API endpoint
  - (the means from which the API can access the resources it needs to perform the task)
  - that responds to requests in real time
- Embedded model
  - Here the model is packaged into an application, which is then published
  - A common example is an application that provides batch-scoring of requests

- What to-be-deployed models consist of depends, of course, on the technology chosen,
  - but typically they comprise a set of code (commonly Python, R, or Java) and data artifacts
    - can have version dependencies on runtimes and packages that need to match in the production environment
    - because the use of different versions may cause model predictions to differ

Model Deployment : Dependency Management

Model Export

- Can export the model to a portable format such as PMML, PFA, ONNX, or POJO
  - improves model portability between systems and simplify deployment
  - helps in reducing dependencies on the production environment

- Come at a cost
  - each format supports a limited range of algorithms,
  - sometimes the portable models behave in subtly different ways than the original

- Whether or not to use a portable format is a choice to be made based on a thorough understanding of the technological and business context.

Model Deployment: Dependency Management(2)

Containerization

- Containerization is an increasingly popular solution
  - to the headaches of dependencies when deploying ML models

- Container technologies such as Docker are lightweight alternatives to virtual machines,
  - allowing applications to be deployed in independent, self-contained environments,
  - matching the exact requirements of each model

- Also enable
  - new models to be seamlessly deployed using the blue-green deployment technique
  - scaling of compute resources for models

- Orchestrating many containers is the role of technologies such as Kubernetes
  - can be used both in the cloud and on-premise.

Model Deployment Requirements

What needs to be addressed – when moving from development to the production

- In customer-facing, mission-critical use cases, a more robust CI/CD pipeline is required

- Typically involves:
  1) 1Ensuring all coding, documentation and sign-off standards have been met
  2) Re-creating the model in something approaching the production environment
  3) Revalidating the model accuracy
  4) Performing explainability checks
  5) Ensuring all governance requirements have been met
  6) 6. Checking the quality of any data artifacts
  7) Testing resource usage under load
  8) Embedding into a more complex application, including integration tests

Model Deployment Requirements(2)

- One thing is for sure is needed:
  - rapid, automated deployment is always preferred to labor-intensive processes
- In heavily regulated industries (e.g., finance and pharmaceuticals),
  - governance and regulatory checks will be extensive
  - are likely to involve manual intervention
- The desire in MLOps, just as in DevOps, is to automate the CI/CD pipeline as far as possible
  - speeds up the deployment process
  - enables more extensive regression testing
  - reduces the likelihood of errors in the deployment

Thank You!

In our next session: