Machine Learning Operations (MLOps) A Meetup

Will Roberts
Data Science Evangelist
Why ML Ops as a Focus?

• SDLC well-defined, but ML lifecycle has different risks and challenges
• ML tends to be difficult to debug because of its probabilistic nature
• Data driven applications can be complicated and unique to each dataset
Why ML Ops as a Focus?

- If we practitioners don't build these enterprise pipelines correctly, the popular services which affect consumers could degrade in the worst possible ways
Pipeline starts with defining the problem and getting the data.

Feature engineering is the act of embedding your expertise & knowledge into your data set.

A model is only as useful as it is consumable.

A model needs versioning, health metrics, and a retirement plan.
Infrastructure
Model Development and Maintenance
Model Deployment and Monitoring
Infrastructure
### Specialized toolkits:
- Rapidly evolving API’s from OSS
- GPU’s
- Language support

### Infrastructure Abstraction:
- Compute access may collide with privacy constraints (HIPAA, PCI-DSS)
- Trained models need somewhere to live and pre-defined network access

### Heterogenous Data:
- Aggregating data means that it’s already been stored and is now accessible

### Supporting Infra:
- Model certification
- Tools for model evaluation, and live tests
- Model Policy:
  - Upgrades
  - Degrades
  - Retirement
Model Development and Maintenance
Initial Training:

• Immediately face problems of computational complexity, scalability, and data governance.

• Feature engineering and data sanitizing still time-intensive

• Models are their own products (data products), so do not couple them tightly to an application.

Model Updates:

• Models are living data products, that reflect the patterns of the data at the time they were trained

• Model versioning

Reuse/Retirement:

• Models are expensive to train in both time and capital. Reuse, don’t isolate!

• We all need to retire at some point when our usefulness has come to an end.
Model Deployment and Monitoring
Deploy to be useful:
• Old information now, but there was a time when models weren’t designed to be consumed.
• Deployment facilitates reuse.
• Reminder: models are their own products.

Validation:
• Validity (measure of choice), throughput, fairness.

Monitoring:
• Production systems require more than just predictive validity.
• Measuring throughput and latency now too

Infrastructure again:
• These requirements for deployment (model hosting, model evaluation, and model maintenance) are either possible or impossible depending on your infrastructure choices.
Pipeline still starts with defining the problem and getting the data.

Feature engineering is still the act of embedding your expertise & knowledge into your data set.

A model is only as useful as it is consumable.
Conduct pre-deploy tests to assess risk before deployment.

Manage operational risk through safe deploy tech.

Continuously monitor for drift.

Drive improvement by learning from model usage logs.
Pre-deploy Test

Assess the performance risk of new model versions

- Test Data $\rightarrow$ Deployed Model $\rightarrow$ Test Accuracy
- Real Traffic $\rightarrow$ Test Accuracy?

Performance Testing

Assess the operational risk of deploying a new model into production.

- Test Set $\rightarrow$ Model $\rightarrow$ Accuracy 98%
- Environment A: Unlabeled Production Data $\rightarrow$ Accuracy ???
- Environment B: Unlabeled Production Data $\rightarrow$ Accuracy ???
Test Oracle

Testing a model on unlabeled production data runs into oracle problem.

Need a test oracle (performance predictor) that can tell us how the model performs on production data.
A Simple Meta-Model (Performance Predictor)

The meta-model learns under what circumstances the model is accurate/inaccurate.

1. Train the model
2. Score the test set with the model and evaluate
3. Construct new training for the meta-model from the model’s scores on test data
4. Train the meta-model
5. Score production data
Deploy
Pre-Deploy Test
Deploy
Monitor
Improve

Reward-based Learning Toolkit
Performance Prediction Toolkit
Active Learning Toolkit

DS & AI Technical Marketing/ June 2020 / © 2020 IBM Corporation
Canary Deployments
Common approach to address deployment risk.

- Only route small portion of traffic to the new canary release
- Closely observe canary performance
- Roll-up or roll-back as needed
Multi-Armed Bandit Primer

A gambler in front of a row of slot machines who has to decide which machines (arm) to play, in which order and how many times.

ML Examples: Epsilon-Greedy, Thompson Sampling, UCB

Simple Multi-armed Bandit API

#Register a new arm
bandit.add_arm(arm)

#Make an arm choice
arm = bandit.choose(choice_id)

#Observe the reward
bandit.choose(choice_id, reward)
Safe Deploy with Multi-Armed Bandits
Learning to choose the best performing model based on observed reward.

Safe deploy interface to the application:
- Observes reward
- Makes safe model routing decisions
- Hides details about the deployed models
How to reliably detect model degradation over time?

Day 10 Traffic → Deployed Model → Day 10 Traffic

Day 1 Traffic → Deployed Model → Day 1 Traffic

Day 1 Accuracy: 92%

Day 10 Accuracy: 62%
Performance Drift

Occurs when the performance of a deployed model in a given window of production data degrades sharply compared to a reference window.

- Reference performance window can be fixed (e.g. test) (A) or dynamic (trailing window) (B)
- Raise drift alert when. \((\text{performance(ref)} - \text{performance(current)}) > \text{drift threshold}\)

![Diagram of Performance Drift]

Test Set

Production Data Stream

(A) Reference Window = Test

(B) Reference Window = Trailing Window

Current Sliding Window

Feature Distribution A

Feature Distribution B

DS & AI Technical Marketing/ June 2020 / © 2020 IBM Corporation
Drift Detection with Performance Predictor

**Test Set**

- Model
  - (A) Reference Performance = Test Results

**Production Data Stream**

- Performance Predictor
  - (B) Reference Performance = Projected Accuracy (trailing window)
- Performance Predictor
  - Predicted current Accuracy

---

Feature Distribution A

Feature Distribution B
Example Drift Detection
Random Forest Model
Pulsar Astrophysics Data set

https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star

Drift detection:
- Relative to fixed reference window (test)
- Alert threshold = 5 percentage points
Active Learning
Improve model by adding new training data

Active learning selector
Selects items from the log that are most informative while minimizing the number of labels that need to be obtained.
Goal-Driven Active Learning

The best selector also depends on the specific active learning goal

Example:
Improving accuracy per class vs. improving accuracy broadly

Example:
Improving coverage (confidence above threshold) vs. improving accuracy
Python library for active learning
Collection of active learning selectors
• Heuristic selectors
• Trainable selectors
• Bias selector
• Meta-selectors
• Evaluation/simulation utilities

Supporting a set of goals
• Improve accuracy
• Expand coverage
• Improve accuracy of select classes
• Reduce bias
• Compound goals, e.g.:  
  • Improve accuracy without increasing bias
  • Improve coverage and reduce bias
ML Ops Summary

AI Builds AI: We developed a set of AI-based lifecycle toolkits to address these challenges by

- Managing operational risk
- Increasing automation & agility
- Driving continuous improvements