Google

Automating Operations with ML

tmu@google.com, stross@google.com

Quick Intro

Agenda

- Intro
- Use cases and problems
  - Why systems operators want to solve problems with ML
  - What primary candidate types of applications exist (in this context)
  - Why those don’t, in general, work well in practice
- Applications of ML to production systems in detail
- Analysis of results: what does work and might work (for you)
- Recommendations for future work

Google

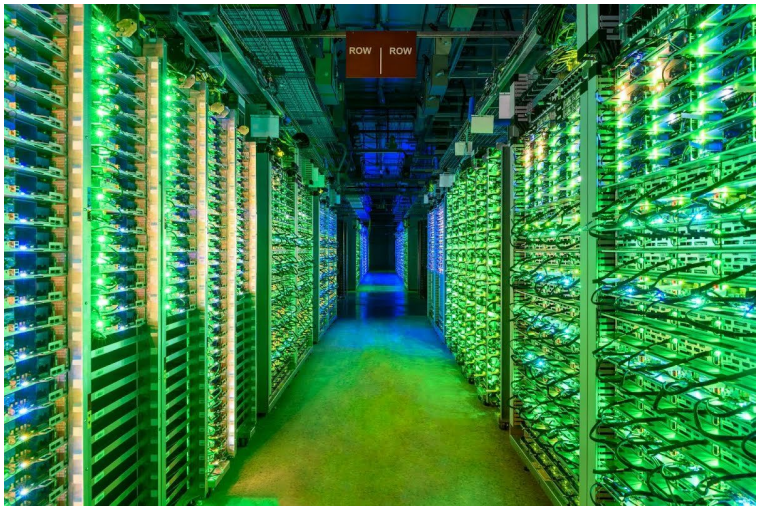
Basic Questions
(phrased as fundamental philosophy problems)

- **Identity**: Who are we? (Basic introductions including, one hopes, amusing anecdotes)
  - Steven: 9 years building ML infrastructure systems at Google. Tech Lead for ML Infra SRE (non-cloud) at Google
  - Todd: 11+ years building multi-tenant ML systems at Google. Leads ML SRE for Google covering internal and external services.
- **Other minds**: Who are you? (In case you do not know; specifically what background you might need to care about this talk).
  - Can spell “ML” and may know somewhat more about it. :-)
  - Somewhat interested in challenges of production engineering
  - Would like to apply ML to improve production computing systems in some way

Basic Questions (cont.)

- **Teleology**: Why are we here? (Not in the universe, rather what problem are we are trying to solve with this talk)
  - There exists a persistent, widespread desire to automate operations with ML
  - “operations”: the activities necessary to maintain the functioning of some (computer) system.
  - There exists the myth that automating operations with ML is easy or common.
  - Very few of these applications work well in practice although a couple sort of do and one works pretty well.
- **Epistemology**: What do we know? (spoiler alert)
  - We know a lot of what doesn’t work (in practice and in production).
  - We know a little about what does work.
  - We have some ideas of what we might all try next.

Context: Operations/Production



Context: large scale computer systems offering shared services.

In that context, “operations” are the activities necessary to monitor, manage and maintain the availability of those services.

“Production” is the environment with the highest availability and reliability requirements.

While many of these observations may apply to other environments (manufacturing, healthcare, law), doing so is beyond the scope of this presentation.

Applications of ML to Operations

Use Cases and Problems

Google

What do systems operators want to do with ML?

Annoying/repetitive/boring tasks (that are still worth doing), typically those that are difficult to automate perfectly using heuristics.

- **Anomaly detection** - in logs usually, possibly with automated alert authoring
- **Automated monitoring maintenance** - updating thresholds and parameters, suppressing bad alerts
- **Capacity Planning/Prediction** - where do we need more capacity in the future, where will we run out?
- **Automatic Service/Resource Scaling** - where do my services need more (or fewer) resources?

Google

What goes wrong

Not enough labeled data, not cost competitive with simpler (heuristic) solutions

- Many of these problems do not have enough (high quality) labeled data.
  - Can't use reinforcement learning without clear rules
  - Semi-supervised approaches struggle because the problems are dynamic, making appropriate behavior look anomalous
- Getting more (high quality) labeled data is going to be expensive
- Some of the problems are only amenable to (what is now) very expensive ML approaches compared to the problem.
- Heuristics often get comparable results for much less effort and lower cost.

T. Underwood, “All of Our ML Ideas Are Bad (and We Should Feel Bad)” https://www.usenix.org/conference/srecon19emea/presentation/underwood

Production Automation with ML

What actually works

Google

How we evaluated ideas for ML

- How we evaluated ideas:
  - Has it been demonstrated in a realistic production use case?
  - Is there enough data to significantly beat heuristics?
  - Can that data be practically labeled?
  - How expensive is it to implement?
  - How reliable is it likely to be?
- Where we looked
  - Google projects we have either proposed or evaluated
  - Broad-but-shallow literature review for implementations of techniques mentioned above
  - **Note: most of the cited research is not Google work.** We're grateful for the excellent work of so many teams.

Anomaly detection from Logs: open research topic

Automatically detecting when something bad happened from logs, flagging the anomalous portion for human review.

Problems:

- **Cost**: Large log files require expensive processing
- **Ordering**: Challenging when systems are parallel
- **Fragmentation**: Full system state spread across multiple files
- **Consistency**: Logs formatted differently across systems
- **Dynamic**: Formats and output patterns change over time, making them inconsistent with each other and previously trained models.
- **Labeling**: Generally requires expert to label a log as anomalous, with limited number of true positives of any particular type, and many possible combinations of variables to train over.
- **Poor S/N ratio for logs (many "anomalies" that are fine)** (this is what makes labeling hard)

Mitigations:

- Consistency and cost can be mitigated with major engineering investment to create ML-optimized "logs" (or monitoring)

N. Bosch, & J. Bosch. (2020). Software Logging for Machine Learning. https://arxiv.org/pdf/2001.10794.pdf

Photo by Pixabay from Pexels

Google

Alert Triage: difficult

Given a set of alerts, determine which ones don't merit human review.

Problems:

- **Consistency**: Alerts are formatted in many different ways by different programs.
- **Dynamic**: Alert frequencies and types shift over time.
- **Effectiveness**: May only eliminate ~2/3 of false positives with minimal loss of true positives. Poor value compared to alternatives.
- **Labeling: Requires skill to label an alert as anomalous or not, with limited number of true positives on a stable system.**

Mitigations:

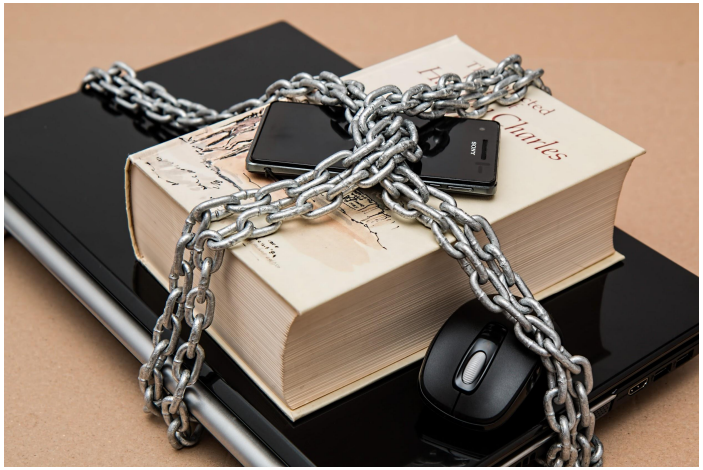
- Standardizing alert format can mitigate consistency.
- Given 1k alerts to train on with active learning (out of a larger set), ~100 true positives may be enough.
- Periodically refresh the model with new data periodically.

M. Bierma, J. Doak, & C. Hudson(2016). "Learning to Rank for Alert Triage". https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/bare_conf.pdf

Photo by Pixabay from Pexels

Google

Security: difficult



Detecting malware, intrusion, and insider attacks that standard heuristics can't catch by flagging binaries, apps, devices, traffic flows, and users.

Problems:

- **Complexity**: Huge number of potentially useful features
- **Dynamic**: Attack types change rapidly (sometimes hours).
- **Data**: Need to have enough real incidents to provide training data. for an ML model. Might not get attacked enough. (!)
- **Labeling: Requires expert to label an attack, with limited number of true positives of any particular type, and many possible combinations of variables to train over.**

Mitigations:

- Build tools for domain experts to search for, find, and analyze incidents, so they can find the rare true positives. This is expensive.
- Once you find a few true positives, use active learning to build a model, gradually finding more.
- Refresh the model with new data frequently to catch new attacks.

D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou (2011). "Detecting adversarial advertisements in the wild". In http://www.eecs.tufts.edu/~dsculley/papers/adversarial-ads.pdf

Google

Photo by Skitterfoto from Pexels

Auto Scaling: easier

Use historical usage patterns to estimate future resource needs and scale capacity up or down to meet SLOs while saving money.

Solved Problems:

- **Effectiveness**: Substantial demand has predictable patterns (ex: diurnal cycles), which allow ML to beat heuristics.
- **Dynamic**: Periodic retraining can adjust for pattern changes. Can trade safety/accuracy off with money.
- **Labeling Data**: Monitoring provides copious automatically labeled data (Is it in SLO + Resource usage & demand)
- **Cost**: Pays for itself easily in many cases.

Weaknesses:

- unforeseen demand spikes will still happen

M. Wajahat, A. Gandhi, A. Karve., & A. Kochut (2016). "Using machine learning for blackbox autoscaling". https://www3.cs.stonybrook.edu/~anshul/igsc16_mlscale.pdf

Image by Pashminu Mansukhani from Pixabay

Key Results

Problems are dynamic requiring periodic model refresh.

Human labeling is expensive compared to value for most applications.

While many ML techniques do solve Operational management problems, they often do so at a cost that is higher than the value that they provide.

One common alternative: straightforward Heuristics. These are brittle and somewhat less flexible, but are still almost as good as ML for many applications.

Of the techniques evaluated, Autoscaling is the most effective application of ML.

Photo by Skitterphoto from Pexels

Future Work

Google

Proprietary + Confidential

Future Work - Data

Folks operating large computing systems are not adopting ML technologies as **effectively or quickly** as we wish we were.

A few reflections from personal experience:

What we planned to do:

1. **Enumerate our problems**
2. **Curate and collect the data (to be ready for ML in the future)**
3. **Use those data to build models and solve problems more easily!**

Proprietary + Confidential

Future Work - Data

Folks operating large computing systems are not adopting ML technologies as **effectively or quickly** as we wish we were.

A few reflections from personal experience:

What we planned to do:

1. **Enumerate our problems**
2. **Curate and collect the data (to be ready for ML in the future)**
3. **Use those data to build models and solve problems more easily!**

What we actually did:

1. **Procrastinate**
2. **Hope**
3. **Complain :-)**

(hard to justify big projects with far-off benefits)

Proprietary + Confidential

Future Work - Data

Folks operating large computing systems are not adopting ML technologies as **effectively or quickly** as we wish we were.

A few reflections from personal experience:

What we planned to do:

1. **Enumerate our problems**
2. **Curate and collect the data (to be ready for ML in the future)**
3. **Use those data to build models and solve problems more easily!**

What we actually did:

1. **Procrastinate**
2. **Hope**
3. **Complain :-)**

(hard to justify big projects with far-off benefits)

What we should actually do

Adopt simple technologies that actually work.

Select modest projects with short-term (small) deliverables. Try to build momentum.

Thank You

Photo by Pixabay from Pexels

Google