Machine Learning for Large Scale Systems

Prof. Walter Kriha

8th November 2018

Agenda

- looking back: Distributed Systems and ML
- Scope and Motivation: The role of ML/AI in large scale systems
- Part 1: Using ML/AI for optimization
- Part 2: Using ML/AI for safe operation
- Part 3: The human side

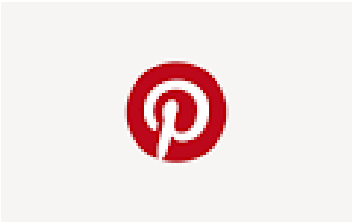
A look back: What distributed systems did for AI...

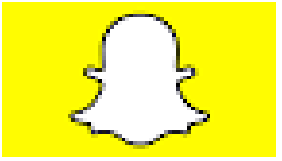
- Distributed file systems for data lakes
- Parallel batch processing for analytics (map/reduce)
- Realtime stream-processing for big data analytics
- Cloud systems for cheap processing
- Integrated special hardware (TPUs, GPUs)

About time to get something back (:-)

Motivation and Scope

Motivation: Scalability and Safety

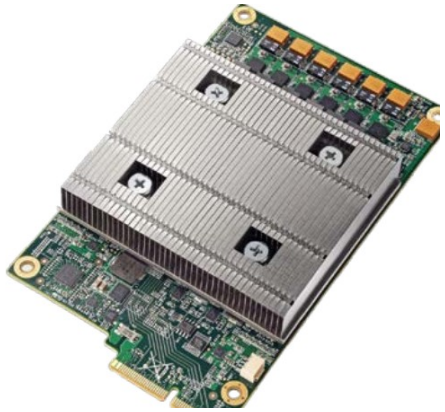




Scope

Systems for ML

- Storage Systems
- Streaming Systems
- Parallel ML (Tensorflow)
- Special ICs
- GPU use for ML



ML in Systems

- Dynamic pricing (AirBnB)
- Feed item ranking (LinkedIn)
- User clustering and churn prediction (Snapchat)
- Search,
- language translation,
- video delivery,
- Walmart: delivery promising, order sourcing, picking/packing Optimization,

….

ML for Systems

- RT Optimization and adaptation
- Data Center operations
- Access optimization
- realtime-learning
- Configuration heuristics
- Resource placement (cache)
- finding/fixing bugs (FB, MS)

...

ML for systems means absolutely mission critical use in large-scale infrastructures!

Questions

What is there to “learn” in large scale systems?

What are the differences to classic system design?

How well do “learning” and dynamic reaction to changes interact?

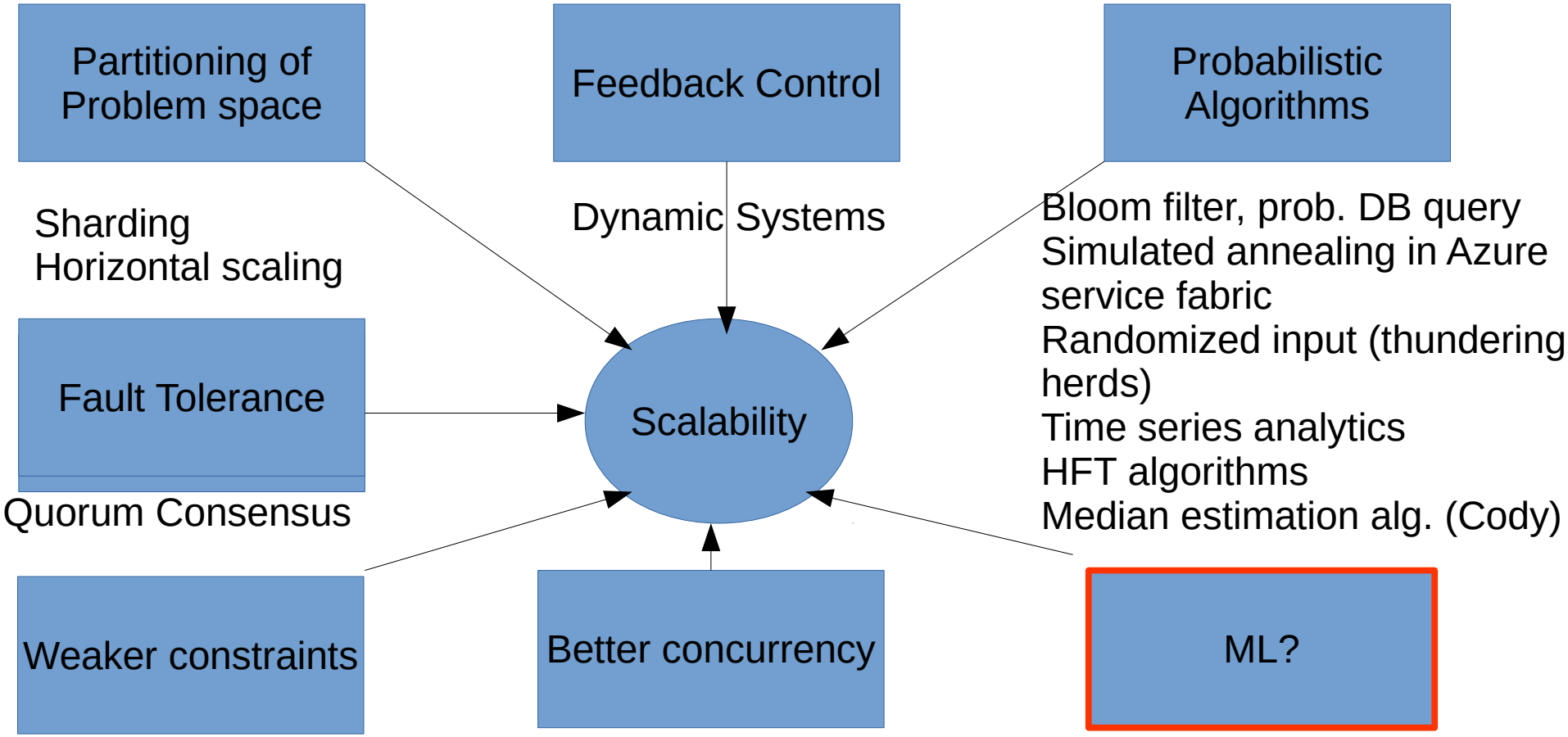
How do we control the effects?

What happens if things go wrong?

Adversarial ML: what can attackers do if they know training data and NN structure?

Part 1: Using ML/AI for optimization

Optimizing Performance and Throughput



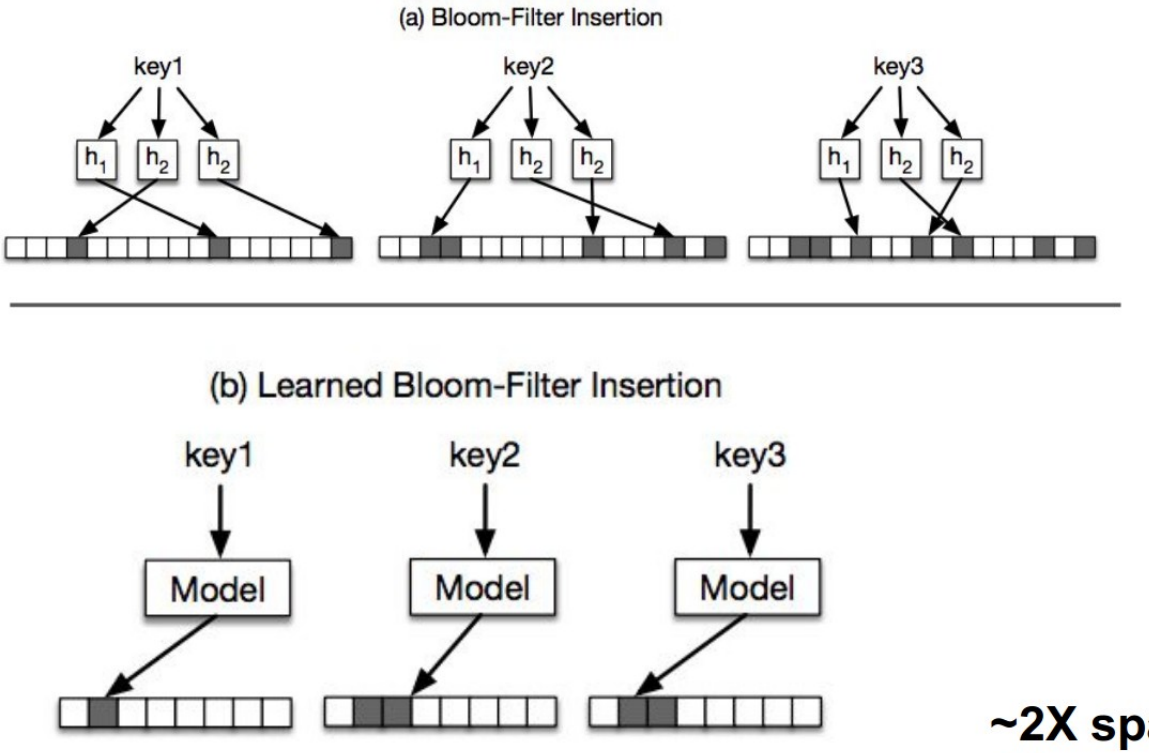
Replacing core components of a system through learned models

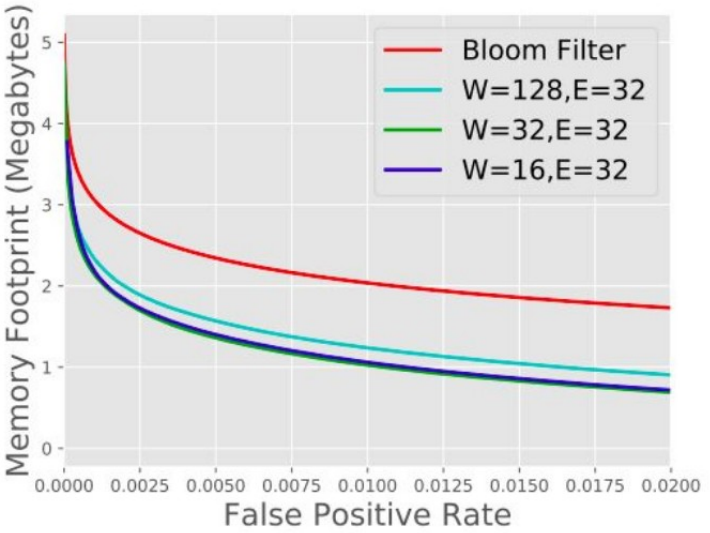
- Access optimization through“Learned Indexes” (B-Trees, Bloom-Filters, Hash-Tables)
- Configuration optimization through learned parameter settings (placements etc.)
- Meta-Learning: design models

Jeff Dean, Machine Learning for Systems and Systems for Machine Learning, http://learningsys.org/nips17/assets/slides/dean-nips17.pdf

Learned Existence Checks

Bloom Filters





Model is simple RNN
W is number of units in RNN layer
E is width of character embedding

~2X space improvement over Bloom Filter at same false positive rate

The Case for Learned Index Structures, Tim Kraska, Alex Beutel, Ed Chi, Jeffrey Dean & Neoklis Polyzotis, arxiv.org/abs/1712.01208

From J.Dean,
Systems for ...

Jeff Dean on Heuristics I

Many programs have huge numbers of tunable command-line flags, usually not changed from their defaults

```
--eventmanager_threads=16
--bigtable_scheduler_batch_size=8
--mapreduce_merge_memory=134217728
--lexicon_cache_size=1048576
--storage_server_rpc_freelist_size=128
...
```

Heuristics have to work well “in general case”

Generally don’t adapt to actual pattern of usage

Generally don’t take into account available context

Kriha: Heuristics have been “learned” once by somebody. They do not adapt. Should they? Should they depend on context? Are there advantages of static heuristics?

From J.Dean, Systems for ...

Jeff Dean on Heuristics II

Compilers: instruction scheduling, register allocation, loop nest parallelization strategies, …

Networking: TCP window size decisions, backoff for retransmits, data compression, ...

Operating systems: process scheduling, buffer cache insertion/replacement, file system prefetching, …

Job scheduling systems: which tasks/VMs to co-locate on same machine, which tasks to pre-empt, ...

ASIC design: physical circuit layout, test case selection, …

Keys for Success in These Settings

(1) Having a numeric metric to measure and optimize
(2) Having a clean interface to easily integrate learning into all of these kinds of systems

From J.Dean, Systems for ...

Jeff Dean on Heuristics III

Meta-learn everything

ML:

- learning placement decisions
- learning fast kernel implementations
- learning optimization update rules
- learning input preprocessing pipeline steps
- learning activation functions
- learning model architectures for specific device types, or that are fast for inference on mobile device X, learning which pre-trained components to reuse, …

Computer architecture/datacenter networking design:

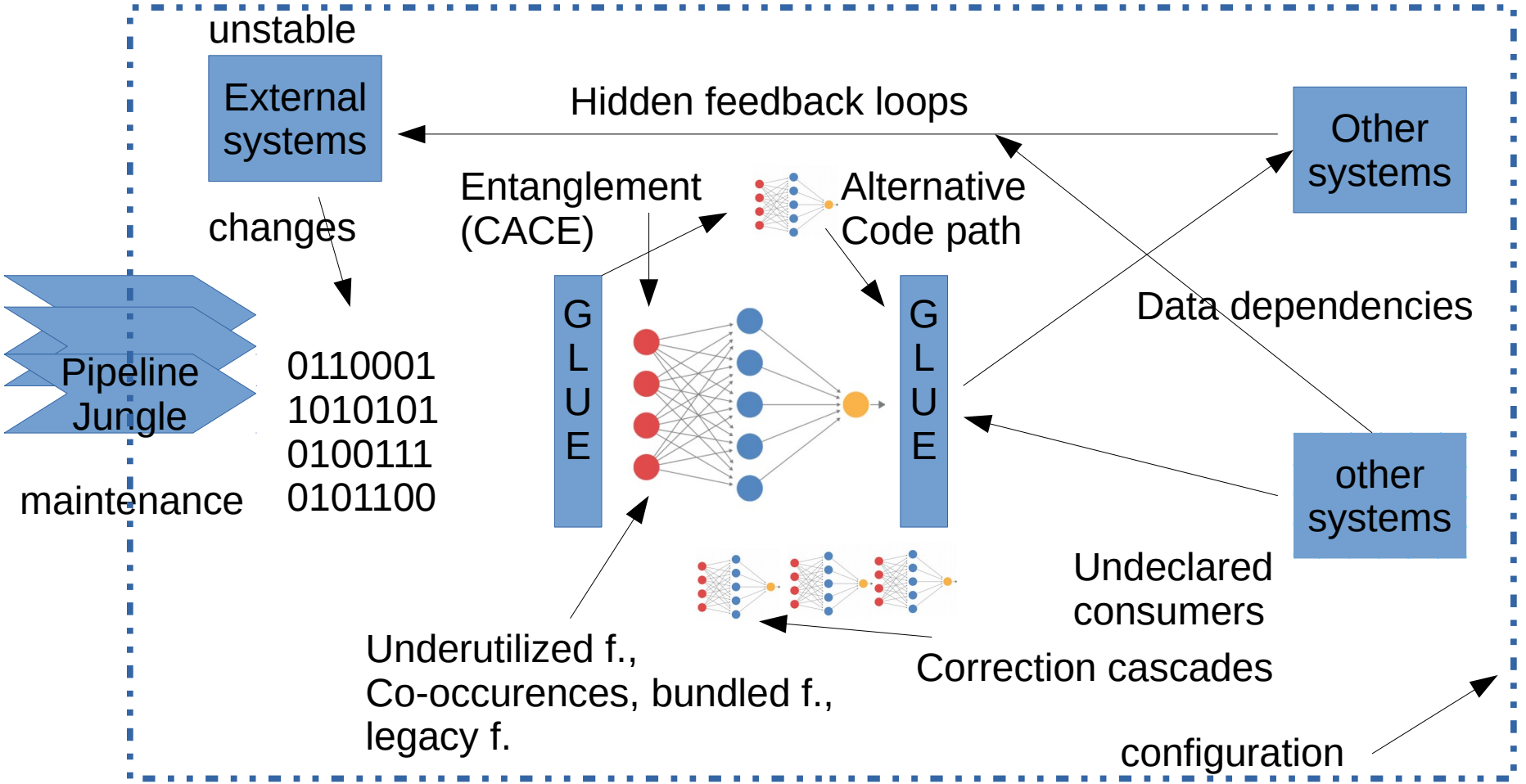
- learning best design properties by exploring design space automatically (via simulator)

From J.Dean, Systems for ...

Is it really so easy?

- system-level interaction between machine learning code and larger systems can create massive maintenance costs (pipeline jungles, dead experimental paths, configuration debt)
- Erosion of abstraction boundaries and decoupling (“data dependencies”)
- re-use input signals in ways that create unintended tight coupling of otherwise disjoint systems
- Machine learning “black boxes”, resulting in large masses of “glue code” or calibration layers that can lock in assumptions.
- Changes in the external world may make models or input signals change behavior in unintended ways – how does one detect changes in predictions?
- Even monitoring that the system as a whole is operating as intended may be difficult (feedback-loops)

Technical Debt in ML

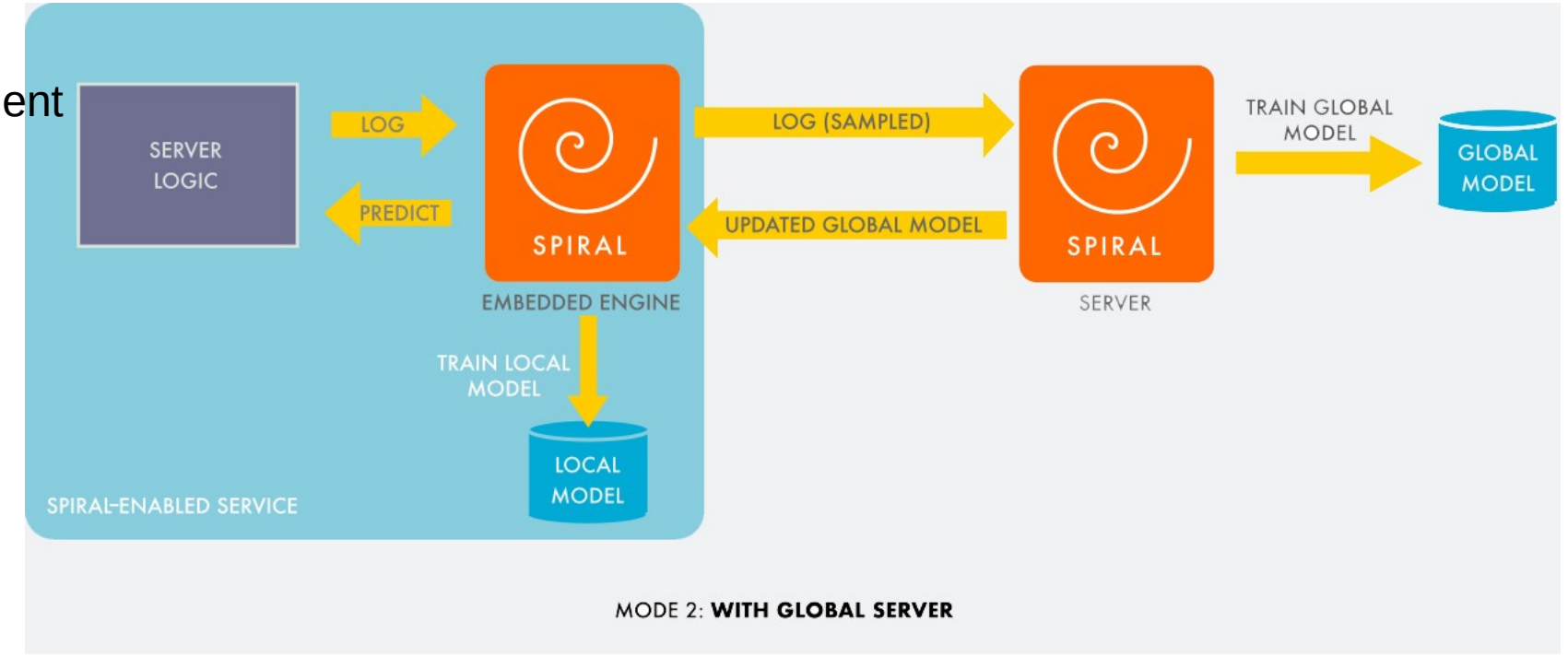


Mitigation: versioned copies, ensembles, deep insight tools, refactoring code paths, checking causality, static dependency analysis etc.

D. Sculley et.al., Machine Learning: The High-Interest Credit Card of Technical Debt (google)

Spiral: a system for self-tuning high-performance infrastructure services at FB scale

Cache placement



Cache object behavior is tracked and new predictions for caching objects are created automatically. If certain types of items are requested less frequently, the feedback will retrain the classifier to reduce the likelihood of admitting such items without any need for human intervention. Previously, blacklists were used to detect spam which took engineers weeks to create – only to become obsolete through changes in queries

Part 2: Using ML/AI for safe operation

System Thinking: A Reminder

- Availability is 99,999 or 99,99999!
- Use algorithms which are widely applicable (general case)
- Use algorithms which are benign (graceful degradation)
- Repair without understanding the cause is useless
- Minimize non-determinism
- Minimize manual intervention
- Provide real-time control of the system
- Self-* properties (self-healing, self-control, etc.)
- Scale across a huge range
- Create observability, explainability
- Carefully track dependencies throughout CI/CD and isolate/decouple components
- Refactor code mercilessly and get rid of technological debt: remove dead code, dead feature flags..

Obviously, there are some contradictions here….

“Why you should not use Google Cloud”

Early today morning (28 June 2018) i receive an alert from Uptime Robot telling me my entire site is down. I receive a barrage of emails from Google saying there is some ‘potential suspicious activity’ and all my systems have been turned off. EVERYTHING IS OFF. THE MACHINE HAS PULLED THE PLUG WITH NO WARNING. The site is down, app engine, databases are unreachable, multiple Firebases say i’ve been downgraded and therefore exceeded limits.

It’s a lonely cloud.

Customer service chat is off. There’s no phone to call. I have an email asking me to fill in a form and upload a picture of the credit card and a government issued photo id of the card holder. Great, let’s wake up the CFO who happens to be the card holder.

We will delete project within 3 business days.

“We will delete your project unless the billing owner corrects the violation by filling out the Account Verification Form within three business days. This form verifies your identity and ownership of the payment instrument. Failure to provide the requested documents may result in permanent account closure.”

What if the card holder is on leave and is unreachable for three days? We would have lost everything—years of work—millions of dollars in lost revenue.

https://medium.com/@serverpunch/why-you-should-not-use-google-cloud-75ea2aec00de

Master Thesis: AI based Intrusion Detection in the Cloud

- clustering algorithms learn in an unsupervised manner
- they group similar samples into clusters
- based on the assumption that anomalies are rare, anomalies either aren’t part of any cluster or they are part of small clusters.
- network characteristics are often discrete and can easily be mapped to a feature space (this is different for system calls)
- the IDS needs to learn online since data streams evolve over time. If new services are deployed into a system the normal behavior changes too, so the clustering needs to be updated with new samples continuously and it needs to forget old data
- the clustering algorithm shouldn’t require repartitioning the complete feature space with each update but only update the affected clusters

Jonas Scheffner, AI based Intrusion-Detection in the Cloud, current thesis (2018) at large corporation.

Summary of Thesis

- generic NIDS for Kubernetes clusters
- cloud poses challenges e.g. to scalability
- Solution based on ORUNADA algorithm:
  - sliding window approach to update feature space
  - dividing feature space into two-dimensional subspaces → solves curse of dimensionality (in high-dimensional feature spaces distances become meaningless)
  - grid clustering on each of those subspaces
  - assigning anomaly scores to points that don't belong to any cluster using the distance to the biggest cluster
  - aggregating scores over subspaces
- evaluation is based on labeled dataset. Labeled data can be acquired using red teams (expensive), honeypots (idle) or by using public datasets
- public datasets can't be representative of the actual environment the IDS is supposed to be deployed to
- solution shows high false positive rate, but recognizes most attacks.

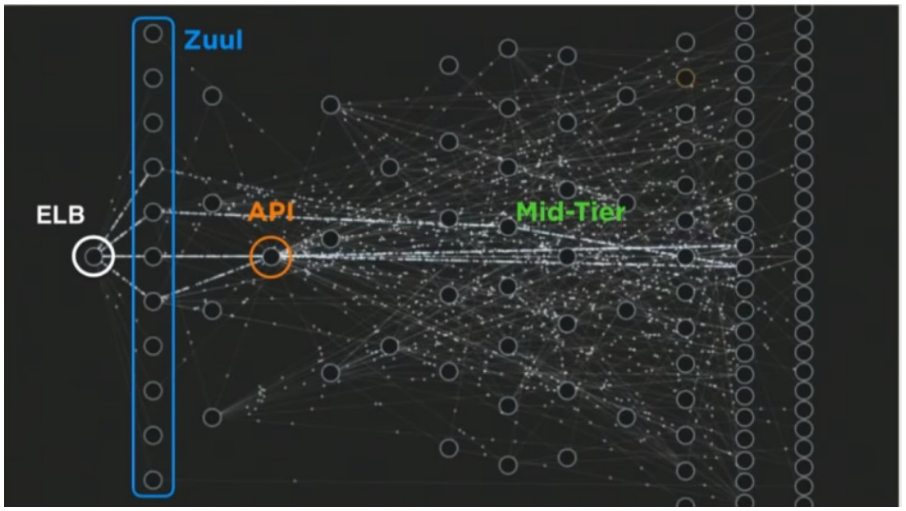
Jonas Scheffner, AI based Intrusion-Detection in the Cloud, current thesis (2018) at large corporation.

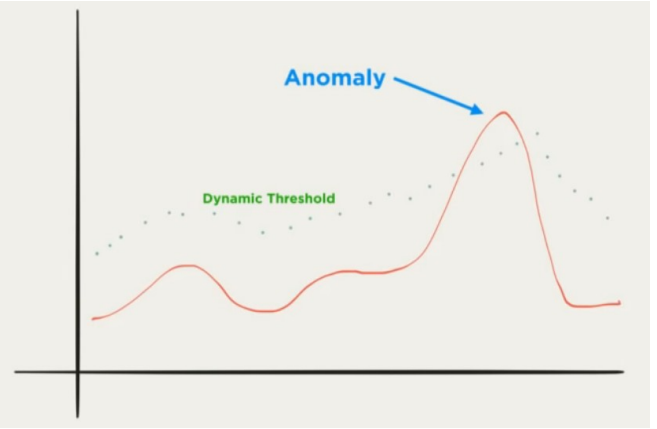
Anomaly Detection: “… with zero AI”

"Scalable Anomaly Detection (with Zero Machine Learning)" by Arthur Gonigberg, Strangeloop 2018

“Anomaly Detection Based on Continuous Monitoring with inspectIT”, Master Thesis 2916, Marius Oehler (CSM)

Netflix Raju





Requirements

Cheap real time analysis
Stable enough to detect sharp spikes
Dynamic enough to adjust to new trends
Lenient enough to filter false positives

"Scalable Anomaly Detection (with Zero Machine Learning)" by Arthur Gonigberg, Strangeloop 2018

Why not ML?

Hard to Train

Comparatively few outages
Very hard to train reinforced model
False positive rate would be very high
Non-stationary data set

Benefits

Small, simple codebase
Predictable behavior
Hand-written rules = high true-positive rate
Cheap processing for thousands of signals

Tradeoffs

Not easily transferable across services
Rules need to be created manually
Alerting decisions need to be tweaked
Need deep operational knowledge

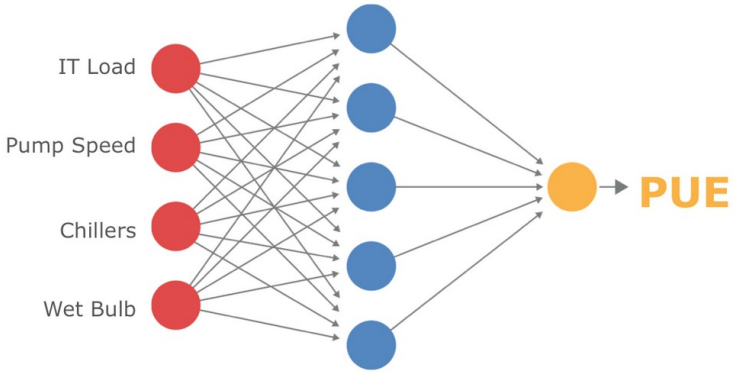
"Scalable Anomaly Detection (with Zero Machine Learning)" by Arthur Gonigberg, Strangeloop 2018

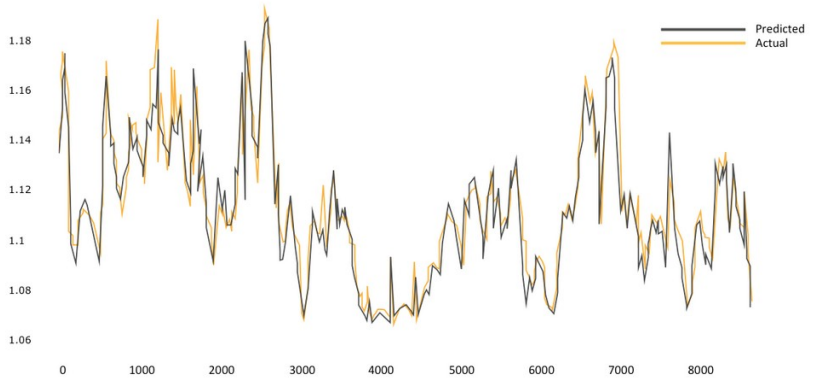
Doing Better: ML controlled data center cooling

1. The equipment, how we operate that equipment, and the environment interact with each other in complex, nonlinear ways. Traditional formula-based engineering and human intuition often do not capture these interactions.
2. The system cannot adapt quickly to internal or external changes (like the weather). This is because we cannot come up with rules and heuristics for every operating scenario.
3. Each data centre has a unique architecture and environment. A custom-tuned model for one system may not be applicable to another. Therefore, a general intelligence framework is needed to understand the data centre's interactions.

https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/

From Simulation to Control



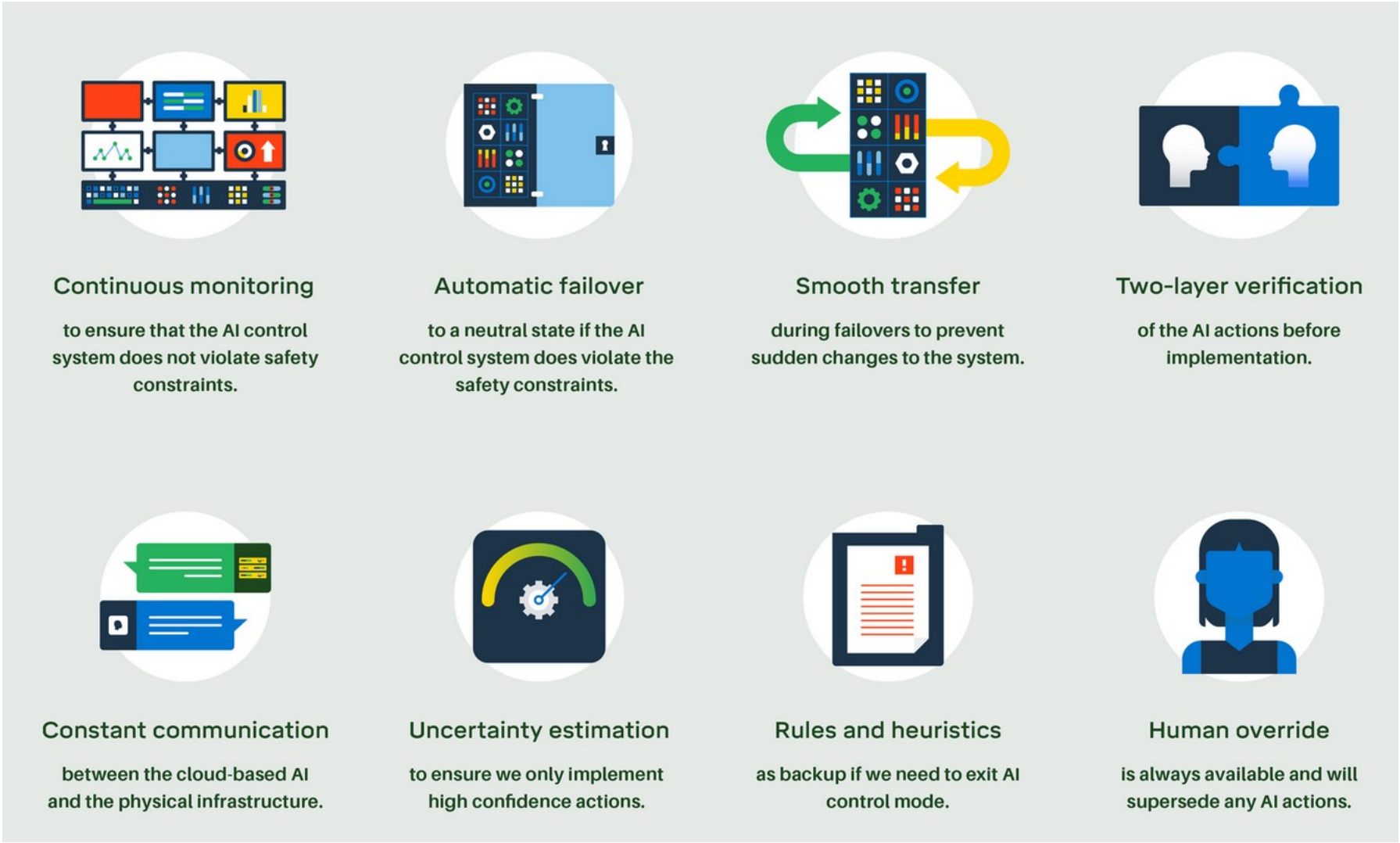


The models can predict PUE with 99.6 percent accuracy.

Reading sensor data in realtime and Calculate an optimal equipment setting

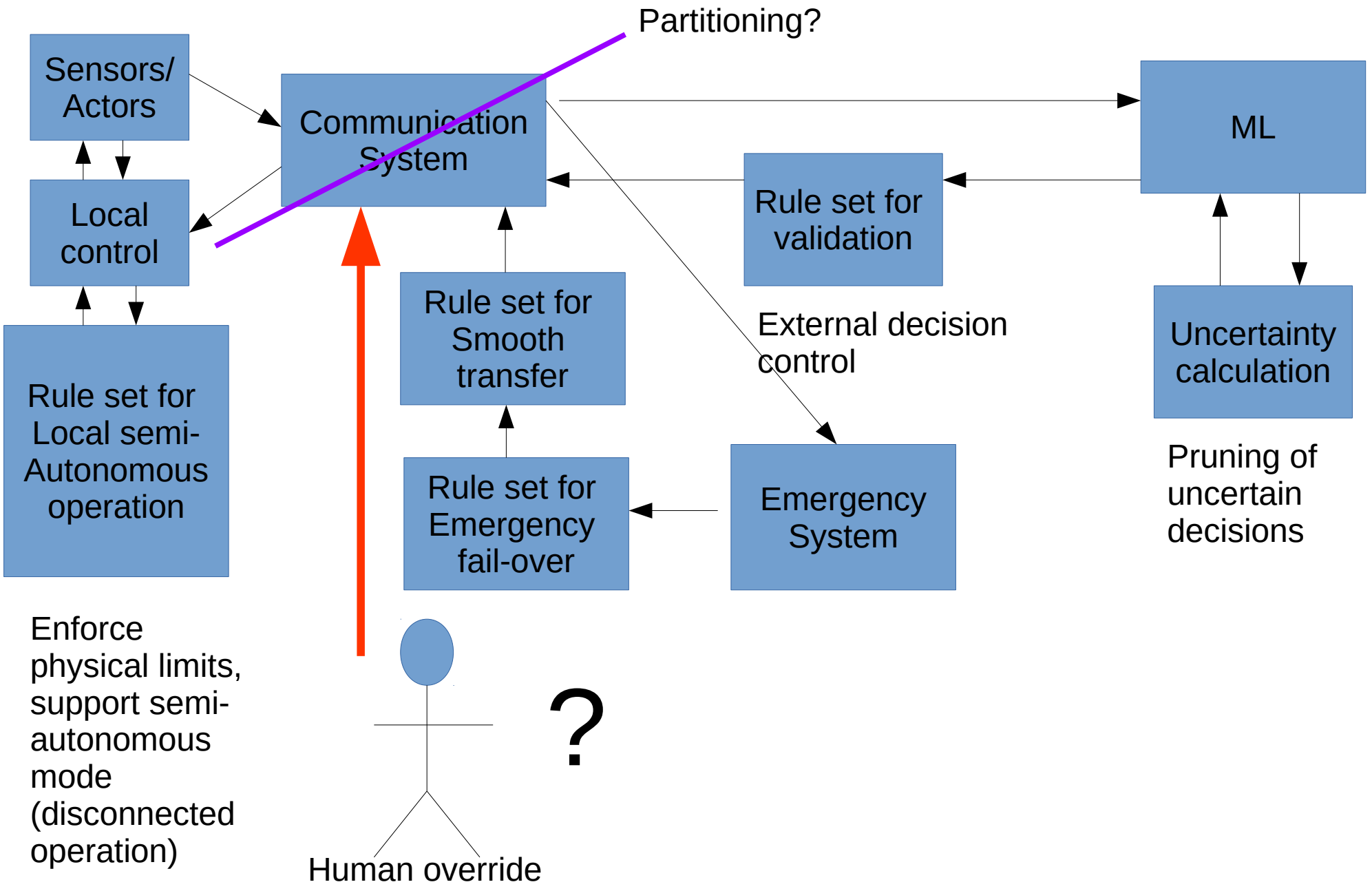
https://googleblog.blogspot.com/2014/05/better-data-centers-through-machine.html

Safety Measures



https://googleblog.blogspot.com/2014/05/better-data-centers-through-machine.html

Interface: ML – System – Human

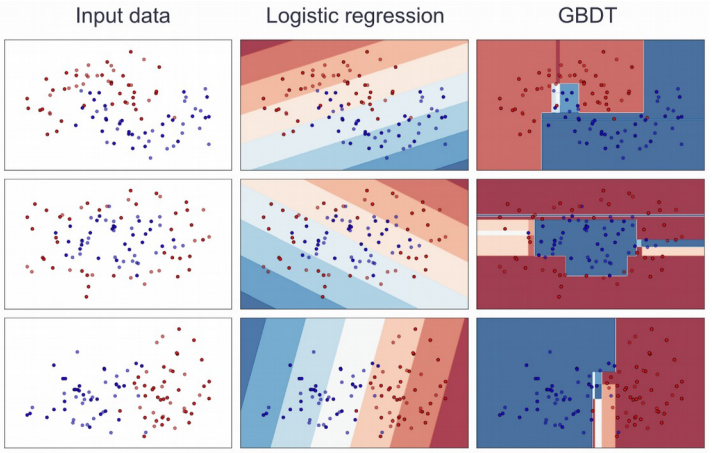


From O(n) to U(n)

- How much aggregated uncertainty is in our systems?
- How big are the possible effects?
- Adaptation and optimization are not automatically goals – they are decisions!
- What is the role of human beings in this whole process – strategic planning or mere troubleshooting?
- Expectations of realtime interventions by humans are nonsense – humans are bad monitors!

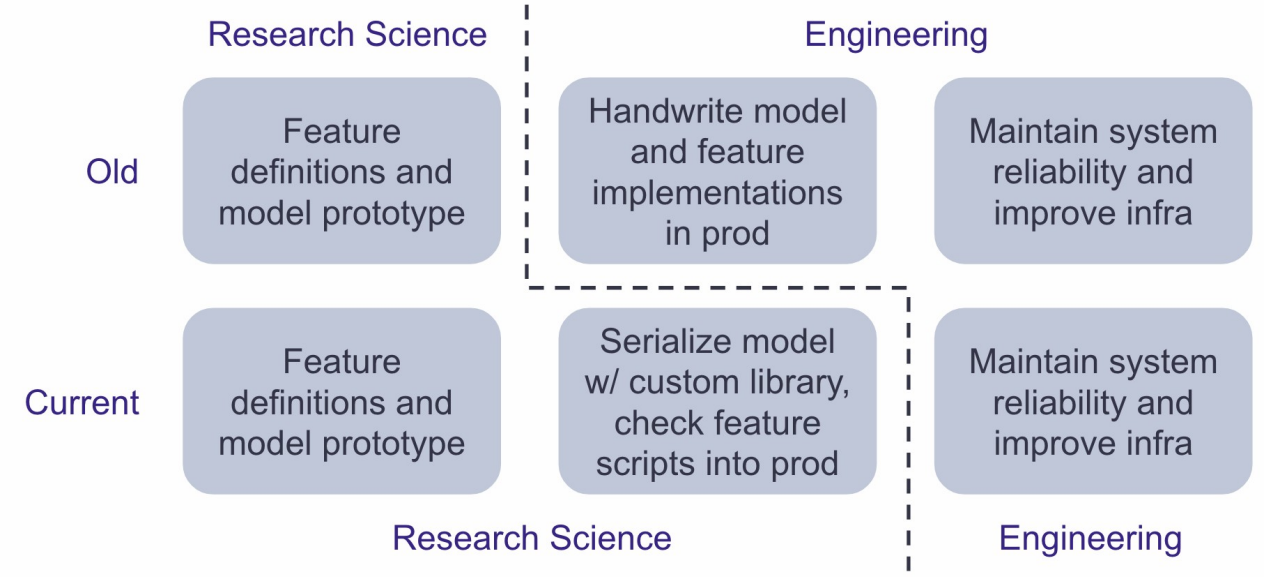
Part 3: The human side

Throwing it over the wall!



Lyft: few/many rides booked already? Suddenly booking of many expensive rides? Initially simple features to indicate fraud. Fraudsters catch up with it (“incubate ride profile”) and require more complex non-linear metrics.

Two problems: Interpretability of ensemble models and problems in production! Solution: get research closer to engineering”



Resources

- Server Scaling Example (elasticity explained with feedback control theory) http://rvanheest.github.io/Literature-Study-Feedback-Control/ServerScaling.html
- Gopal Kakivaya et.al., Service Fabric: A Distributed Platform for Building Microservices in the Cloud
- David H. Wolpert, The Lack of A Priori Distinctions Between Learning Algorithms
- Hyunho Yeo et.al. Neural Adaptive Content-aware Internet Video Delivery
- John Hennessy and David Patterson, A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development, Stanford and UC Berkeley 13 June 2018 https://www.youtube.com/watch?v=3LVeEjsn8Ts
- Orca: differential bug localization in large-scale services Bhagwan et al., OSDI'18
- Data Science In Walmart Supply Chain Technology, https://medium.com/walmartlabs/data-science-in-walmart-supply-chain-technology-bdb5d6b4105c
- Hao Yi Ong, From shallow to deep learning in fraud, A Research Scientist's journey through hand-coded regressors, pickled trees, and attentive neural networks, https://eng.lyft.com/from-shallow-to-deep-learning-in-fraud-9dafcbcef743
- D. Sculley, et.al., Machine Learning: The High-Interest Credit Card of Technical Debt, ghttps://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43146.pdf
- Vladimir Bychkovsky et.al. Spiral: Self-tuning services via real-time machine learning, https://code.fb.com/data-infrastructure/spiral-self-tuning-services-via-real-time-machine-learning/