Software Testing

Can Machine Learning save us?

Marius Liaaen
Carl Martin Rosenberg

Who are we?

“Certus Center for software validation and verification SFI” - hosted by Simula Research Laboratory and sponsored by the Norwegian Research Council

Partners: ABB, Cisco, Kongsberg Maritime, Kreftregistret, Esito

Cisco Systems Norway

400+ employees at Lysaker

Develops meeting room and personal video systems.

This talk

What are the main problems in testing today?

To what extent can Machine Learning address these problems?

What solutions exist? How do they work?

What will the future look like?

Testing today

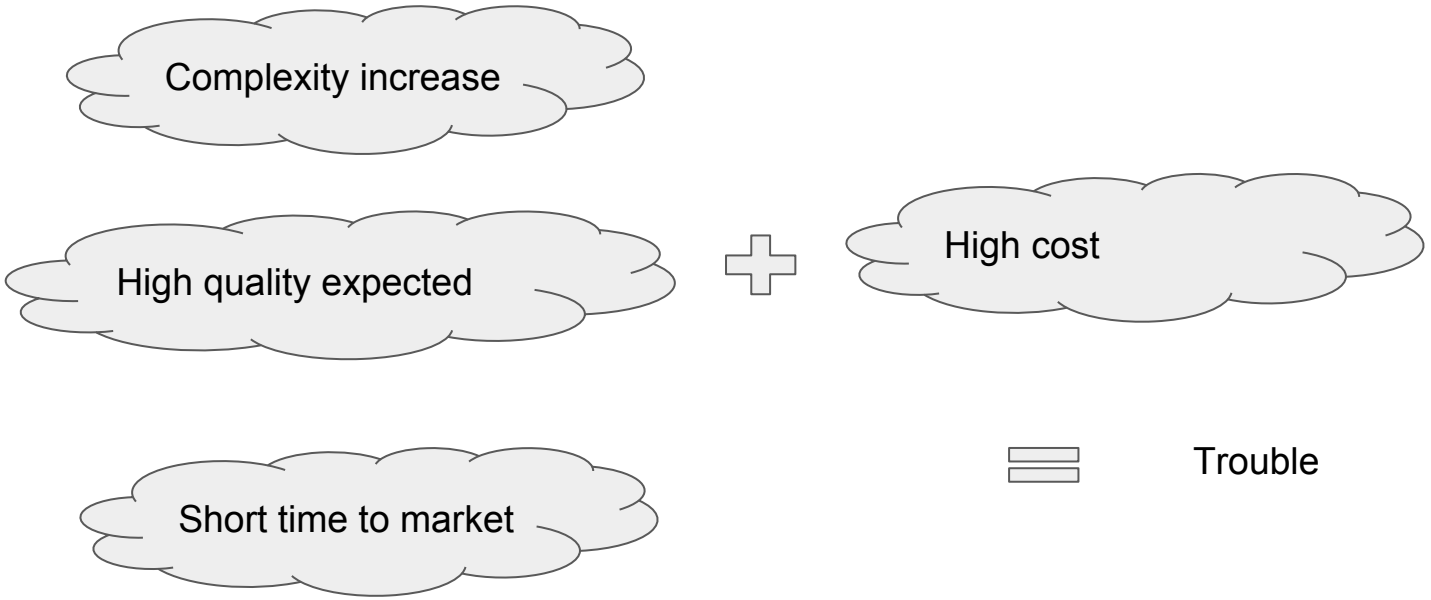
What is software testing?

Understanding the intent of the software and verify that it works as intended.

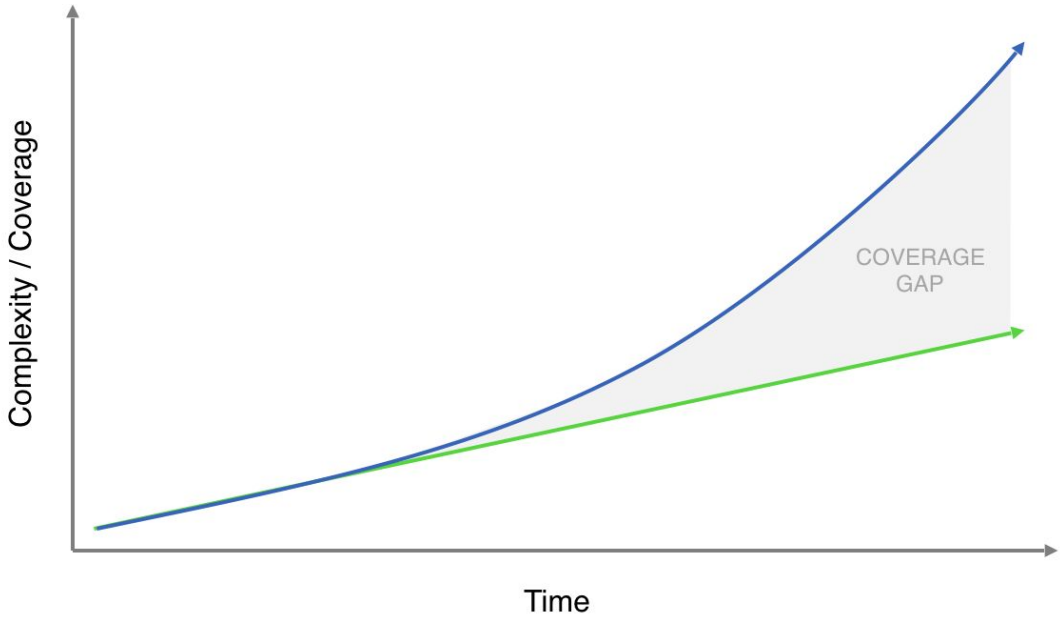
Creating test cases and execute them.

Report issues found and get them fixed.

The problems with testing today



THE COVERAGE GAP IN SOFTWARE TESTING



Features

Complexity increases exponentially as new features and states interact with existing features

Tests

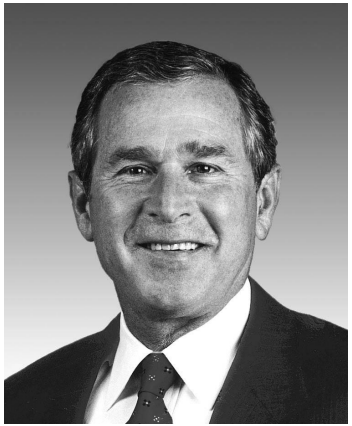
Test coverage grows linearly because they can only be added one at a time

Graph by Jason Arbon:
https://medium.com/app-quality-and-testing/ai-for-software-testing-44052eb0d834

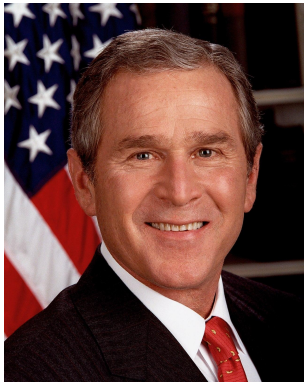
The promise of Machine Learning for testing

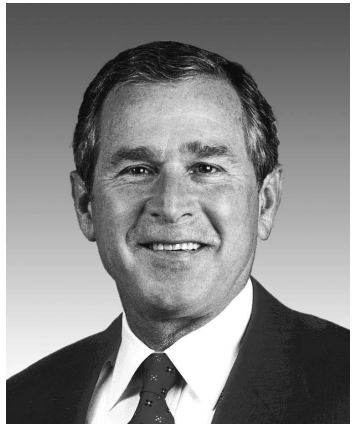
- Testing has stood still for many years, but things are moving on the horizon
- Testing is expensive, and any improvements are worthwhile
- Easier access to the data and tools that enable ML

Why are people using Machine Learning?

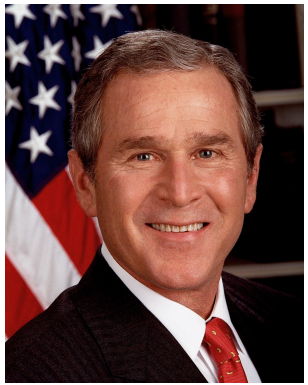


[The Obama-Biden Transition Project, Wikimedia]





[The Obama-Biden Transition Project, Wikimedia]



[A] large portion of real-world problems have the property that it is significantly easier to collect the data (more generally, identify a desirable behavior) than to explicitly write the program.

Andrej Karpathy, Director of AI at Tesla

[https://medium.com/@karpathy/software-2-0-a64152b37c35]

[M]achine learning is concerned with the question of how to construct computer programs that automatically improve with experience

Tom Mitchell, *Machine Learning*

Via machinelearningmastery.com/what-is-machine-learning/

Supervised Learning	Learning from examples
Unsupervised Learning	Finding structure in data
Reinforcement Learning	Learning from rewards

Supervised Learning	*“Here is a picture of a dog”*
Unsupervised Learning	*“Point x looks like an outlier”*
Reinforcement Learning	*“I got 50 more points than last time. Let’s continue doing this!”*

Supervised Learning	Better GUI-testing
Unsupervised Learning	Finding anomalies in test results
Reinforcement Learning	Simulate users, explore programs

Techniques are often combined!

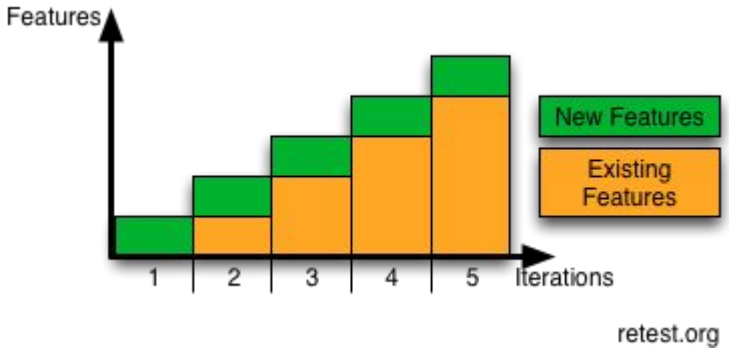
Case 1: Can we drain the swamp of GUI-testing?

Problem: GUI testing is a mess

- Small product changes breaks tests
- Very high maintenance cost
- Little reuse
- Too few checks

GUI Regression testing

Problem:



Cannot craft tests for everything!
Exact pixel matching is NOT the solution!

Solution:

Detect changes in screens (components) from baselines

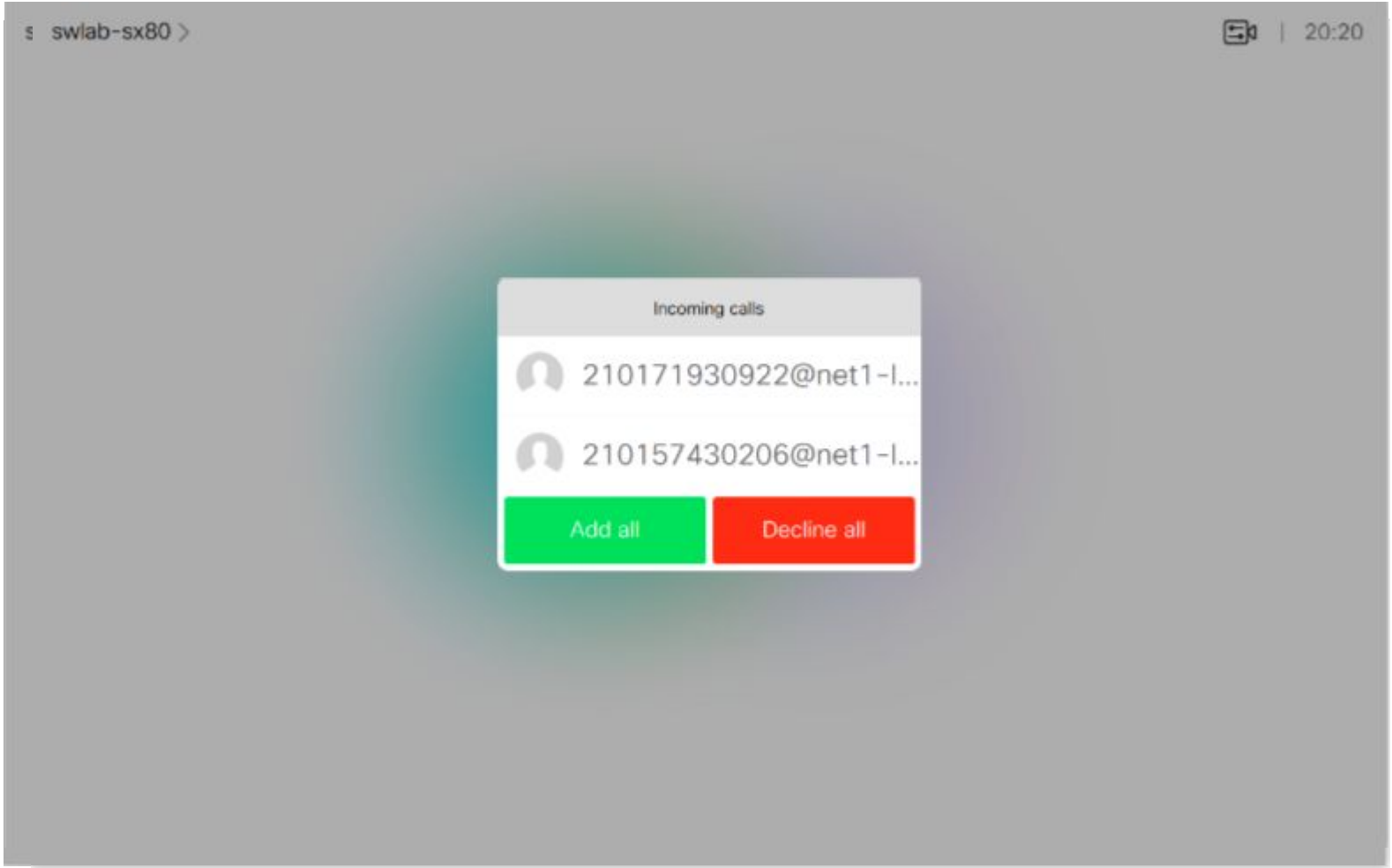
Train to find suspect differences.

Allow more levels of differences

Tools:

Applitools, Chromatic, Retest ++

applitools eyes



Test abstractions in GUI-testing

Problem:

Tight coupling between tests and application

- Tests become flakey
- GUI is “always” changing
- Scenarios lives long
- It’s painful: avoid it -> embarrassing errors

Solution:

- Create long lived, high level tests
- Let the machines find the details
- Train to recognize components using screen shots and/or DOM (self healing)

Tools:

Test.ai (apps), testim.io (web), mabl (web), retest.de (web) ++

Example: abstract GUI testing

Test:

1 make a call

2 enter a number

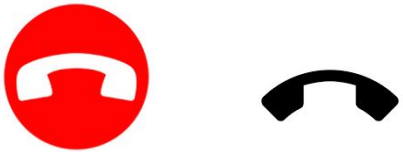
2 disconnect the call

ML finds the GUI elements

Call buttons:

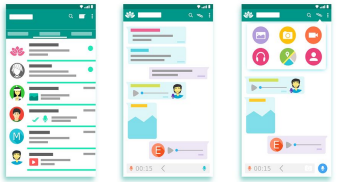
Enter number fields:

Disconnect buttons



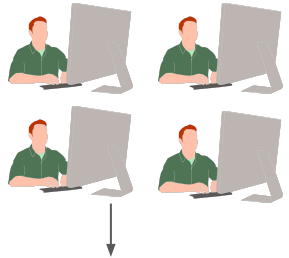
GUI-abstraction with Supervised Learning

Collect screenshots

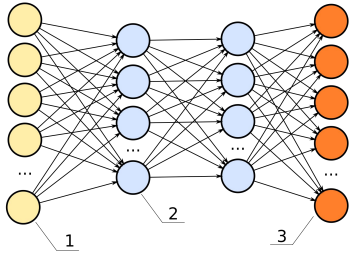


+++

Label images with relevant UI categories



Train ML model on labeled screenshots



Use predicted UI category to perform relevant action

Supervised Learning: Criteria for success

- You have, or can obtain, a sufficiently large labeled data set
  - Often very costly
  - Can sometimes leverage pre-trained models
- Problem can be construed as predicting to a fixed list of classes

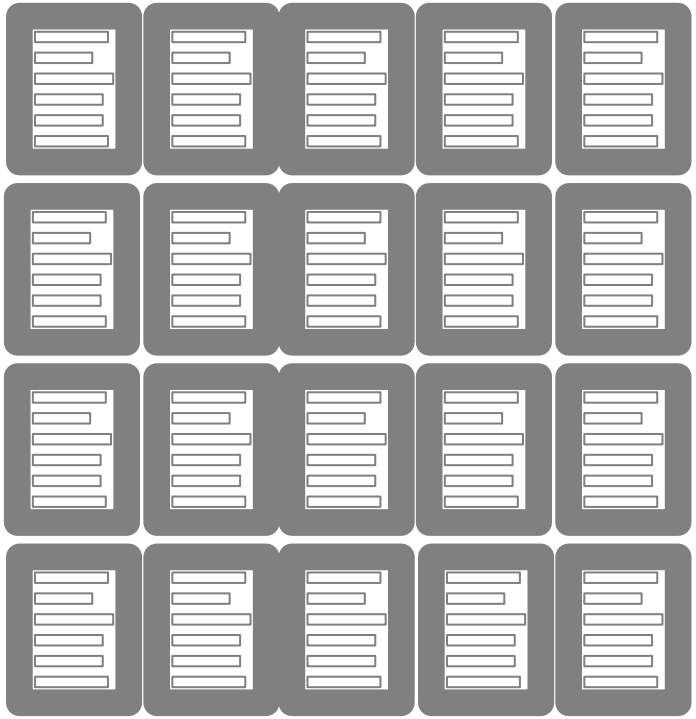
Case 2: Analyzing Test Results in a smarter way

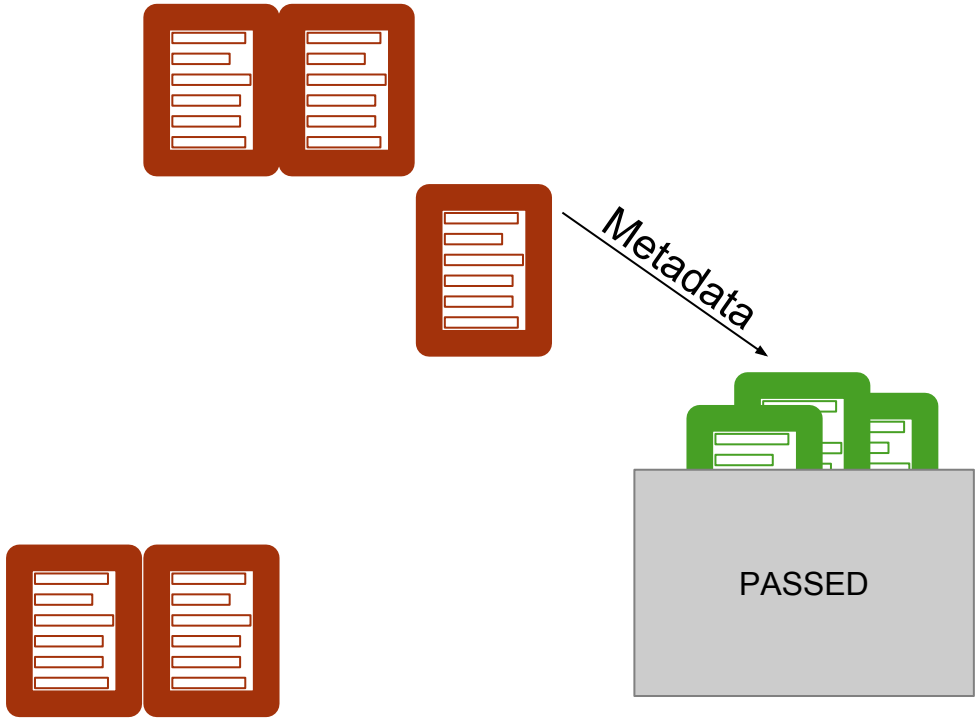
Analyzing test results: Beyond pass/fail

- Fuzziness in testing is increasing
- Difficult to get precise results under all conditions

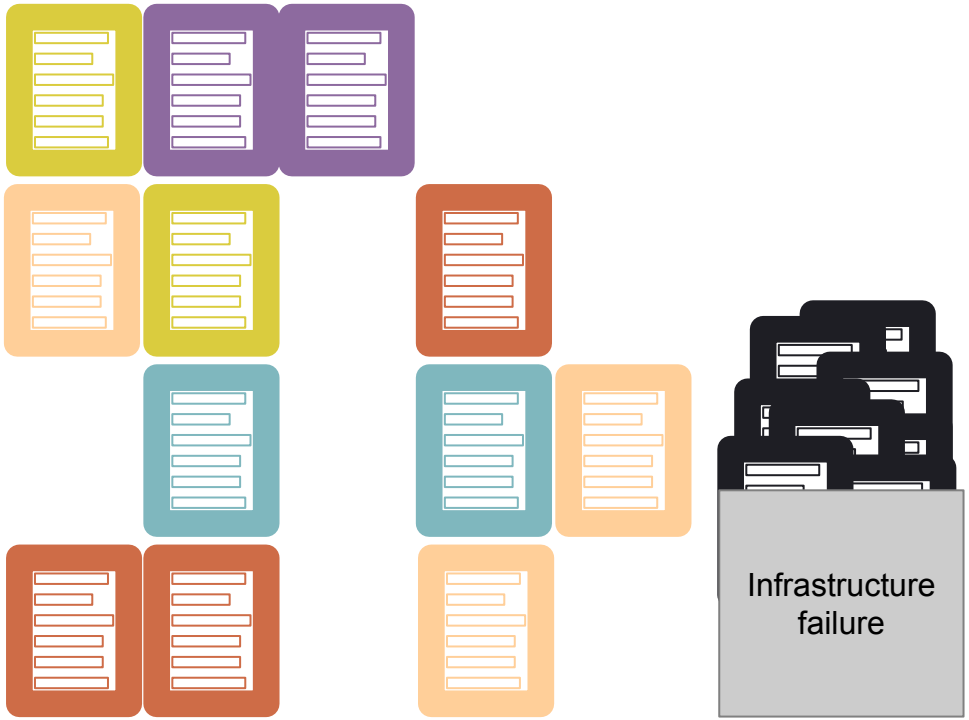
Lots of test results requires manual inspections.

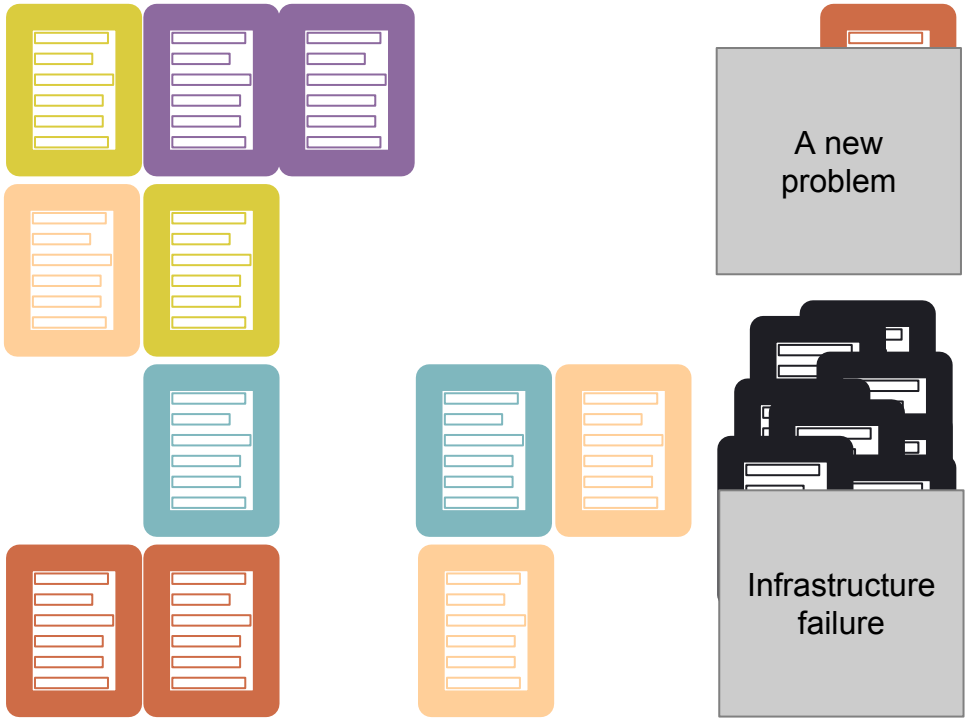
Very time consuming and boring.



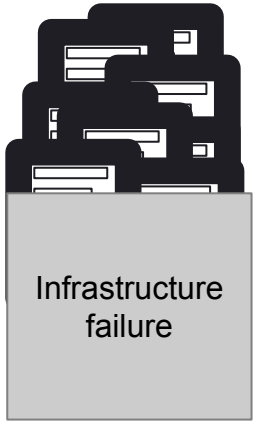

Metadata
PASSED

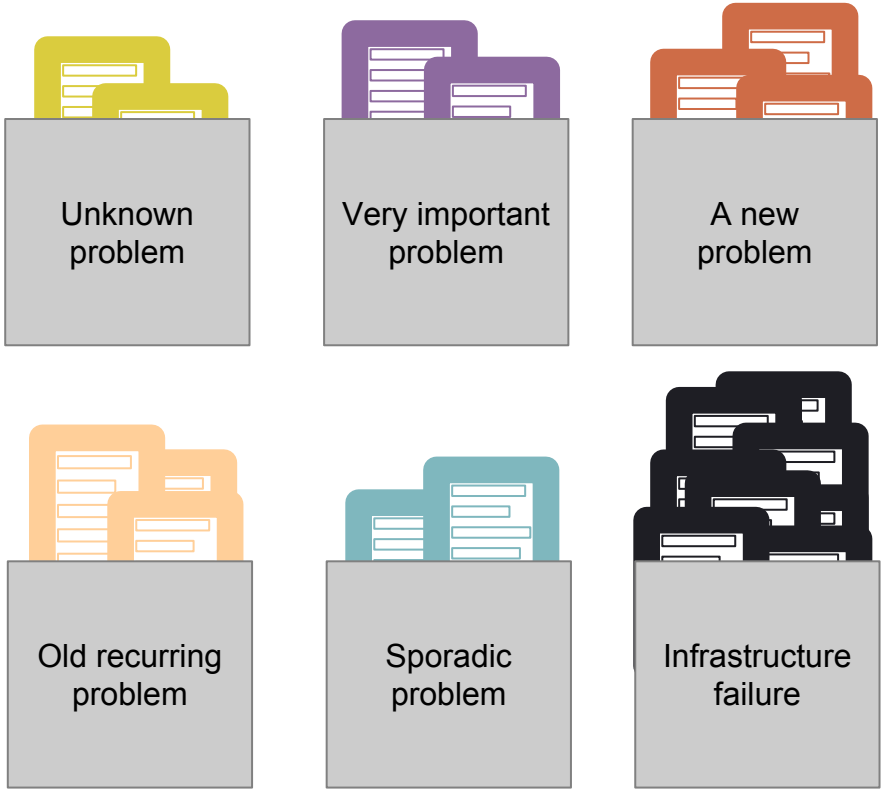

Infrastructure
failure

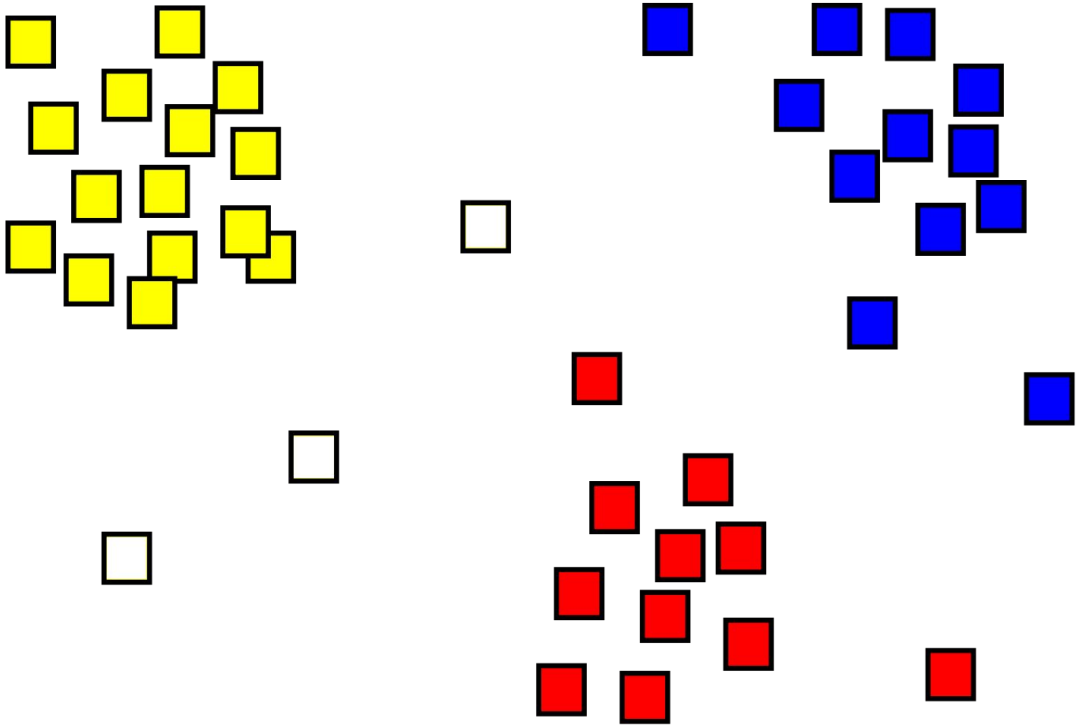

Infrastructure
failure


A new problem
Infrastructure failure

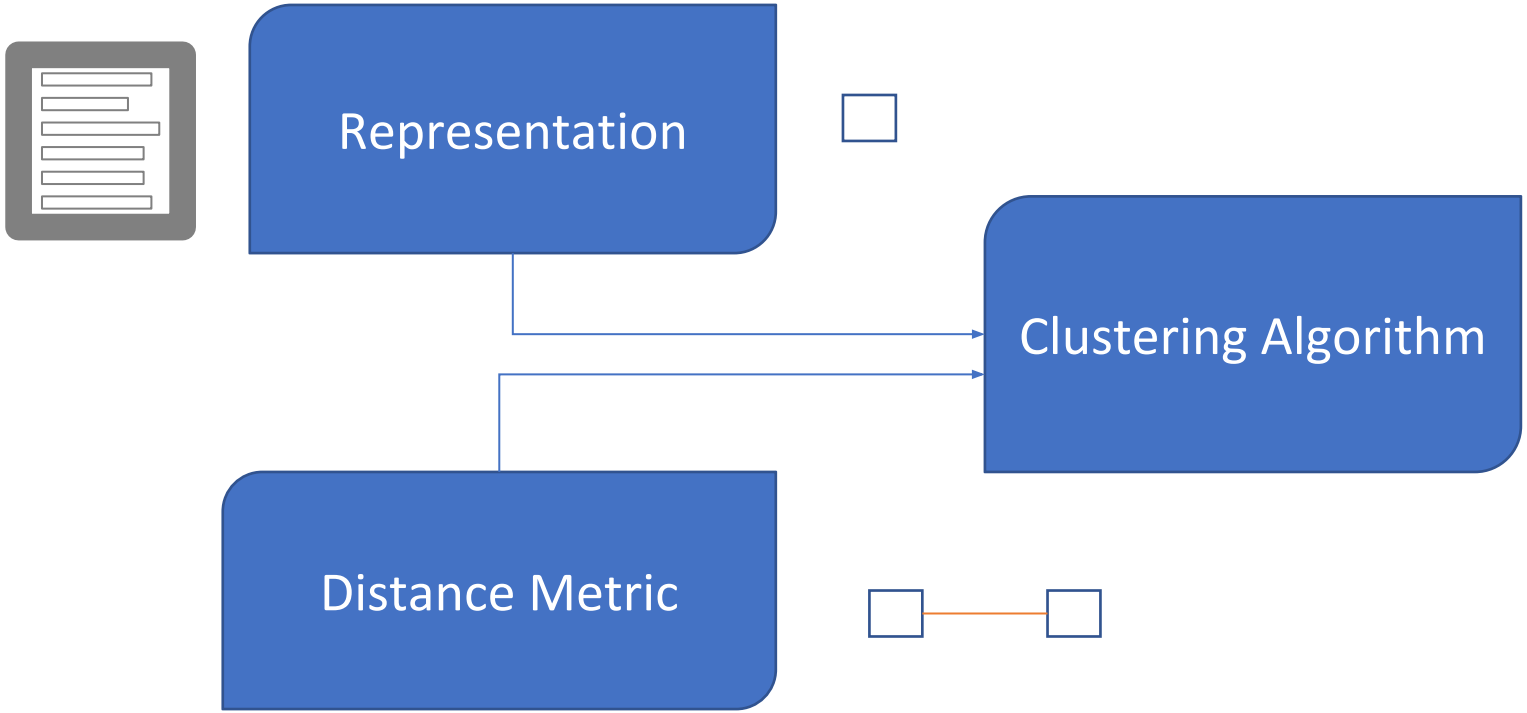

A new
problem


Infrastructure
failure

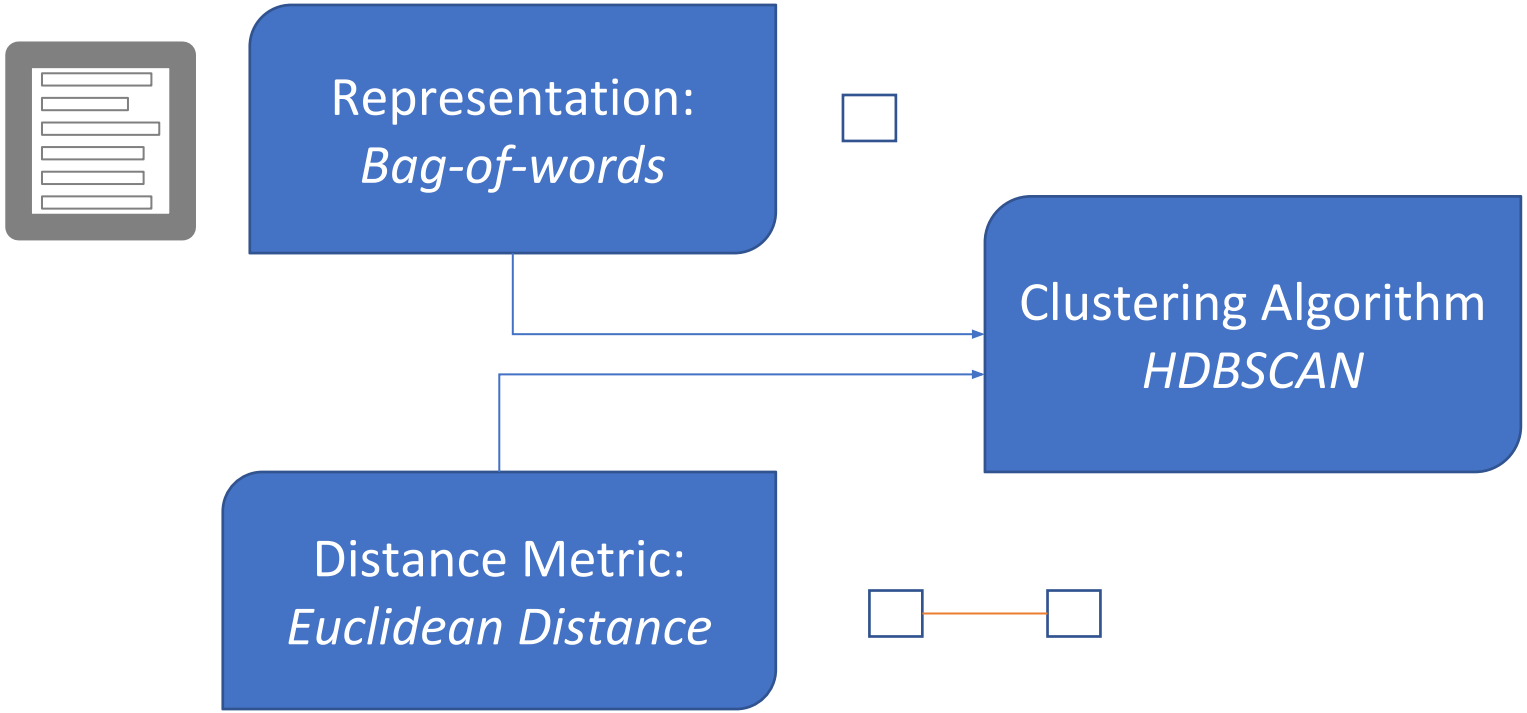

Unknown problem
Very important problem
A new problem
Old recurring problem
Sporadic problem
Infrastructure failure



The ingredients of a clustering solution



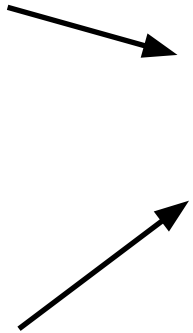
Initial system:



The bag-of-words model

I want more chocolate, mom!

My mom and dad want more sleep

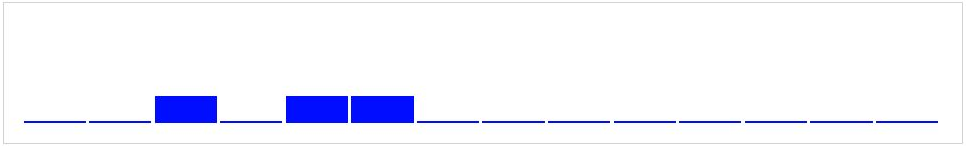


AND	CHOCOLATE	DAD	I	MOM	MORE	MY	SLEEP	WANT
0	1	0	1	1	1	0	0	1
1	0	1	0	1	1	1	1	1

CUCM Ad-hoc conferencing Edge AST

Results in group #1 of 3

Contains 4 of a total of 9 evaluated results.



Results | Comments (0) | Metadata | TFAT (0) | ACR (0)

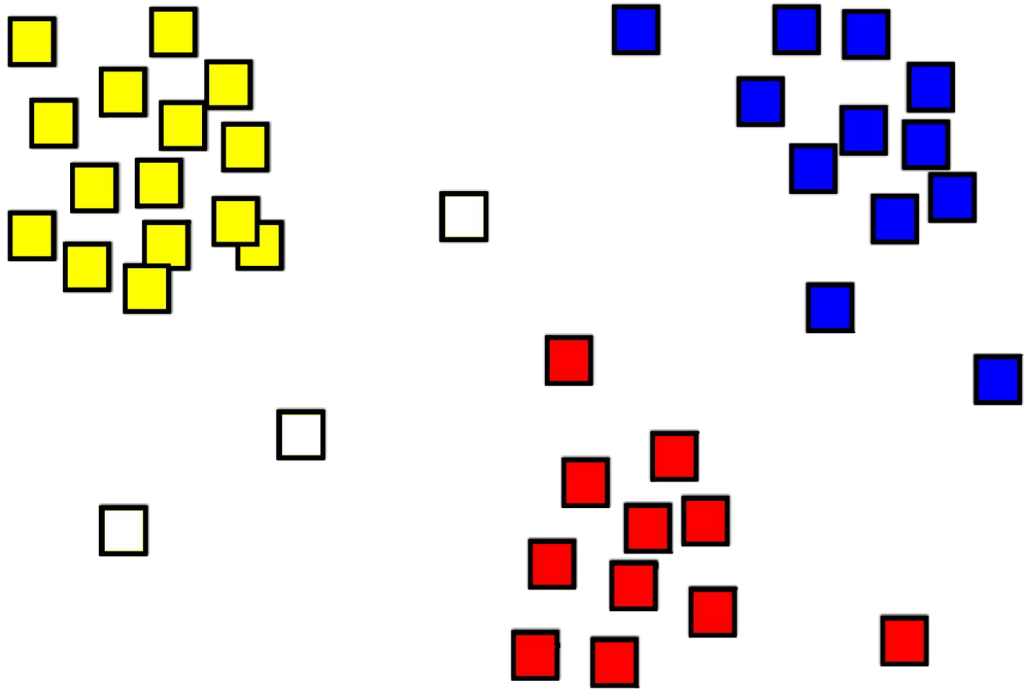
Sort by: Timestamp

NEXT

	Commit	Finished at	Target model	Success	Failure	Duration	Scrubbed	
☑	d0feea2	9.8.2018, 12:45:13	sx20	1	3	981	false	Logs
☑	0f33be9	8.8.2018, 15:41:59	sx20	1	3	886	false	Logs
☑	578f621	7.8.2018, 18:32:27	sx20	1	3	598	false	Logs
☑	2415cb8	2.8.2018, 18:34:38	sx20	1	3	740	false	Logs

Characteristics of Unsupervised Learning

- Analyzes the data without reference to a set of labels
- Looks at internal structure and correlations in the data
- Paradigmatic examples: *Clustering* and *Anomaly Detection*

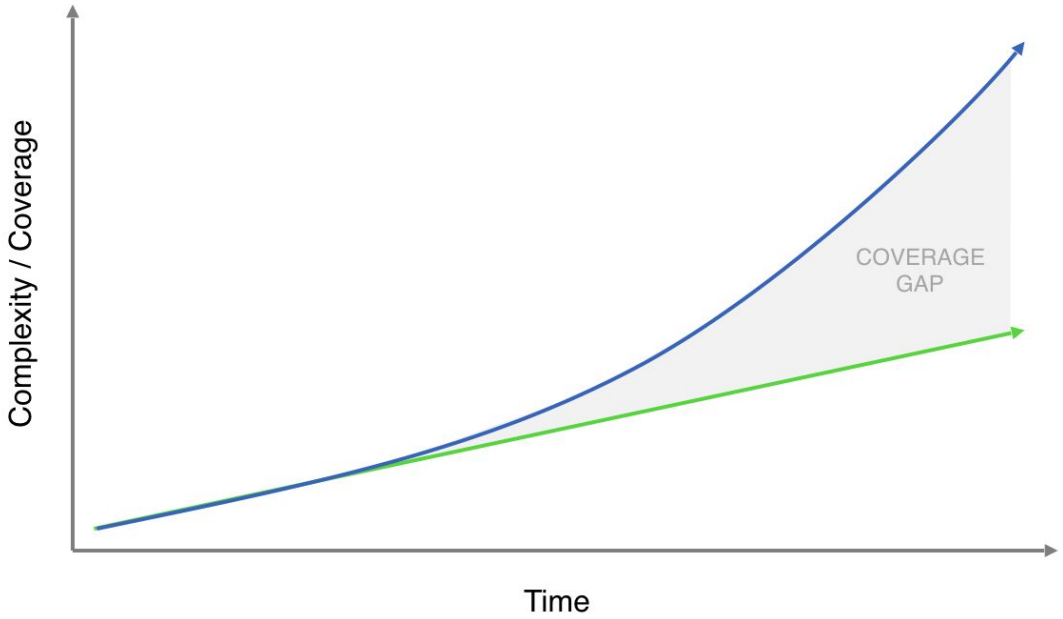


Unsupervised Learning: Criteria for success

- Use on a problem where you want *more perspectives* rather than *predictions*
- Find a suitable evaluation strategy
- The paradox:
  - Cannot be evaluated without a ground truth
  - If you *had* a ground truth, you would likely get better results with Supervised Learning

Case 3: Test more, with less work?

THE COVERAGE GAP IN SOFTWARE TESTING



Features

Complexity increases exponentially as new features and states interact with existing features

Tests

Test coverage grows linearly because they can only be added one at a time

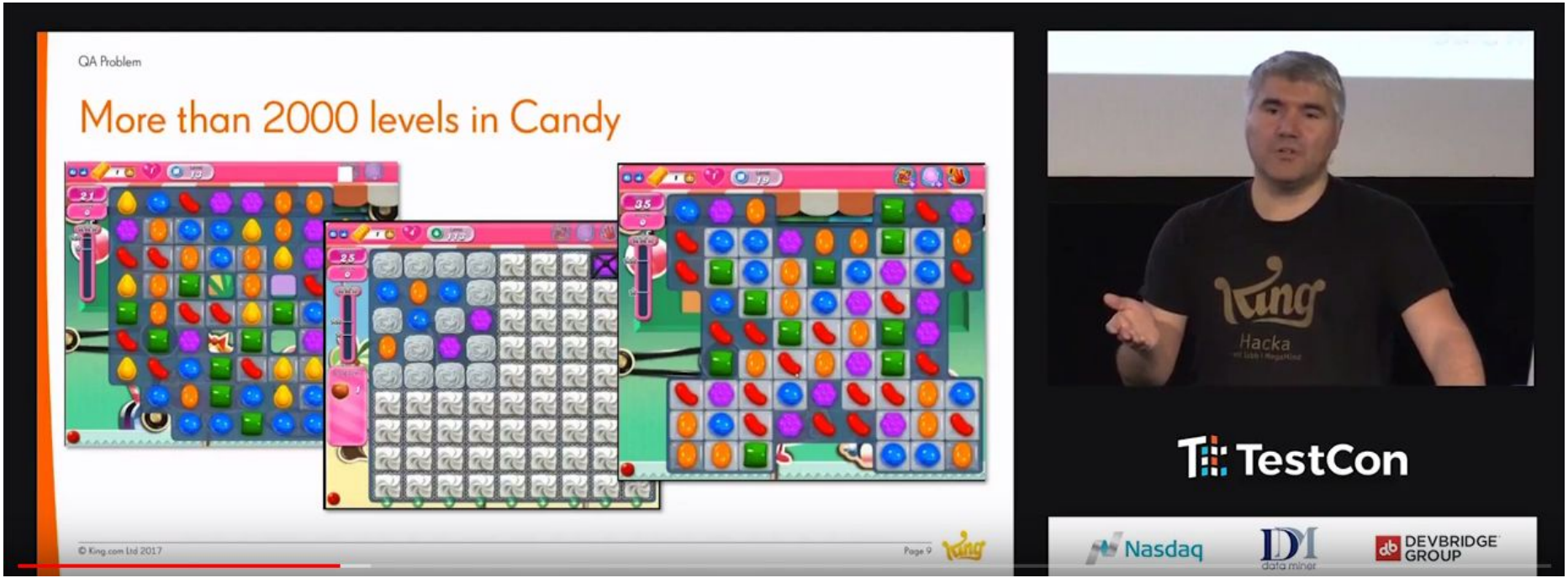
Graph by Jason Arbon:
https://medium.com/app-quality-and-testing/ai-for-software-testing-44052eb0d834

You can never make all the test cases you need.

Can the machines do (some of) the work for us?

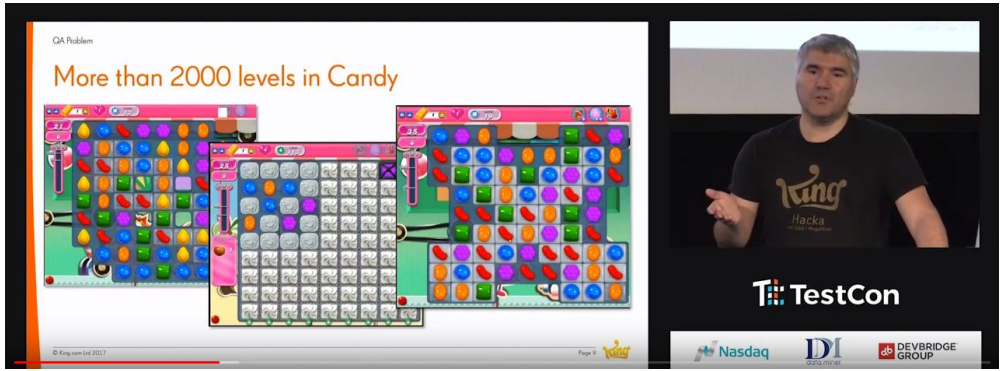
Testing Candy Crush with Reinforcement Learning

Alexander Andelkovic's talk: https://youtu.be/42B-O6TK1bg

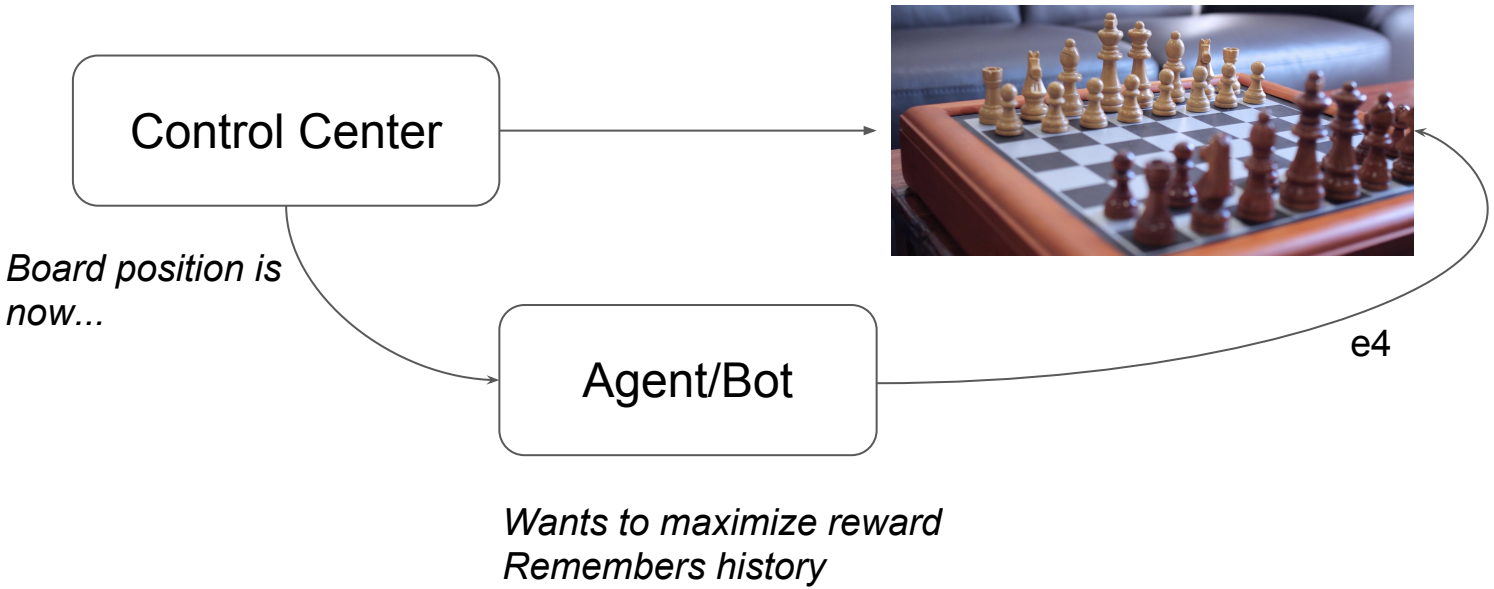


Testing Candy Crush with Reinforcement Learning

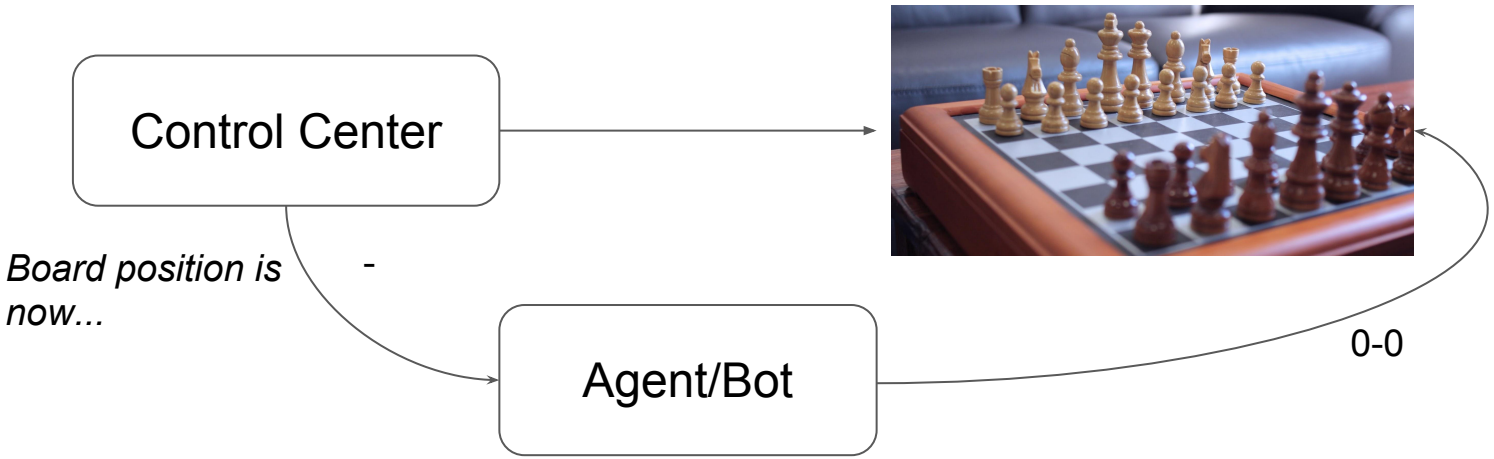
- Created a bot that tries to win at Candy Crush
- Benefits:
  - Check that levels work
  - Check level difficulty
  - See if program can crash
  - Check performance



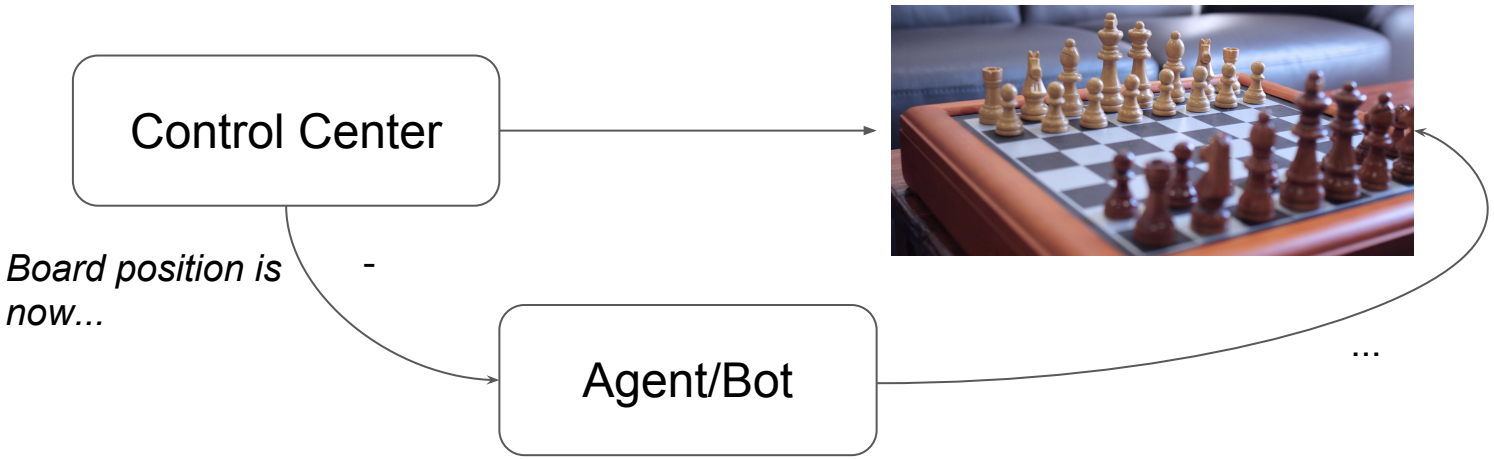
Reinforcement Learning



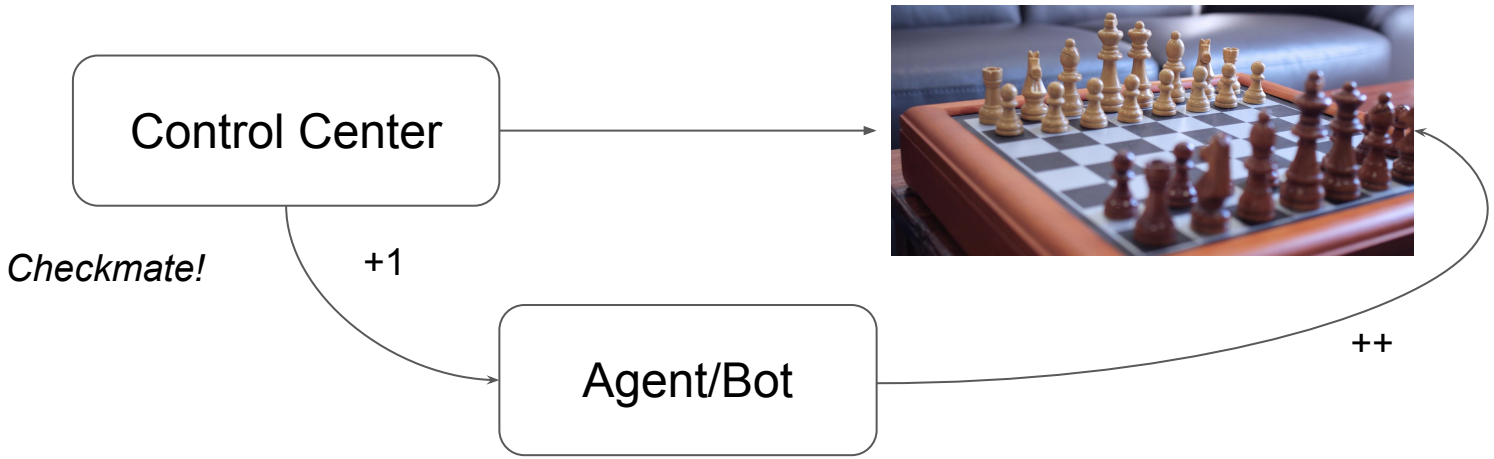
Reinforcement Learning



Reinforcement Learning

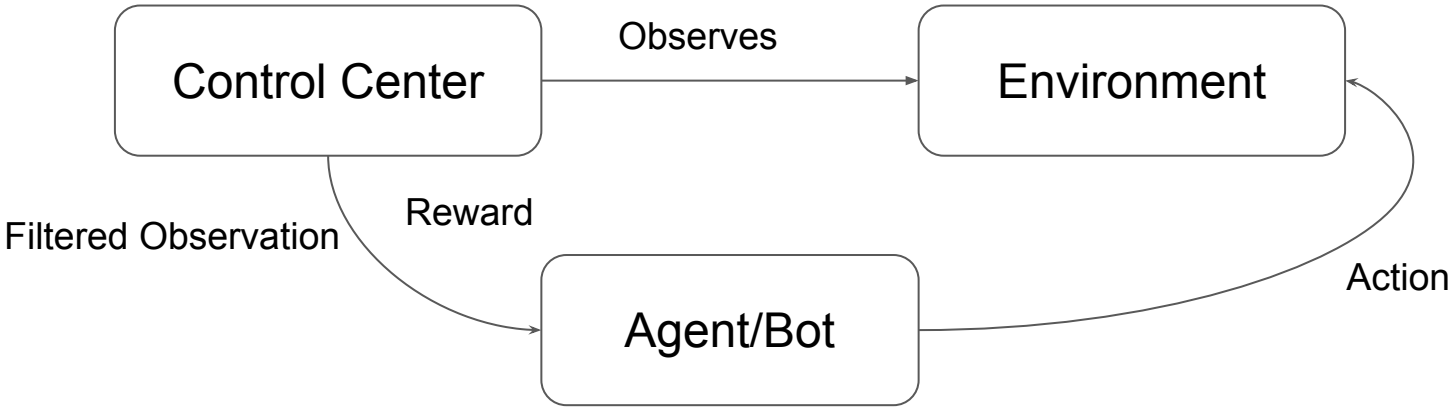


Reinforcement Learning



Iterate thousands of times until a satisfactory performance is consistently achieved

Reinforcement Learning

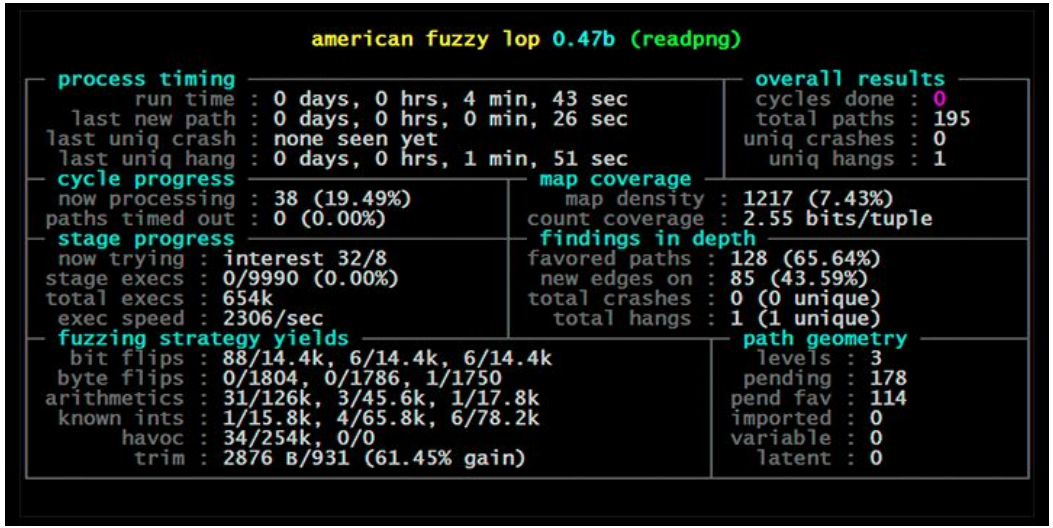


Wants to maximize reward
Remembers history

How to succeed with Reinforcement Learning

- Must have clear goals that can be represented in a reward structure
- Observability and speed
- System Under Test must support rapid re-runs
  - App will have to run thousands, perhaps millions of times for proper training...
  - Being able to save and resume state from an application state is very beneficial

Related: Fuzzing



Other cases

Problem: Helping Developers

“Which tests are relevant to run and who should review this code change?”

Nobody knows everything. Help me do my job.

Learning the history:

- code, test coverage, defects, developer actions etc.

Goals:

- Provide the most relevant information at any stage.
- Suggest improvements in code & test

Problem: efficient test execution

“Too many tests, too short time. Which ones shall I run?”

Learning:

- from previous test results, product changes, coverage

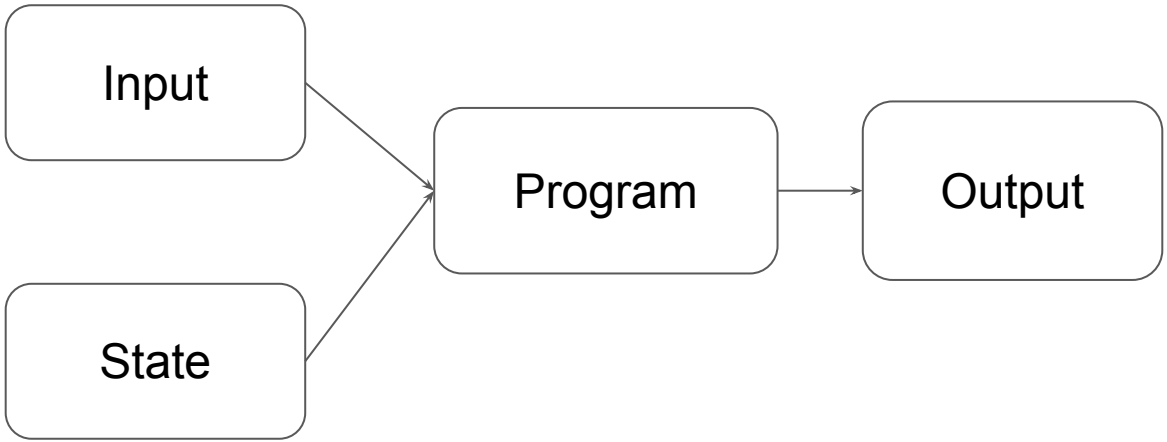
Goals:

- An optimal set of tests needed for a specific product change
- Best possible test suite with constraints (time, resources)

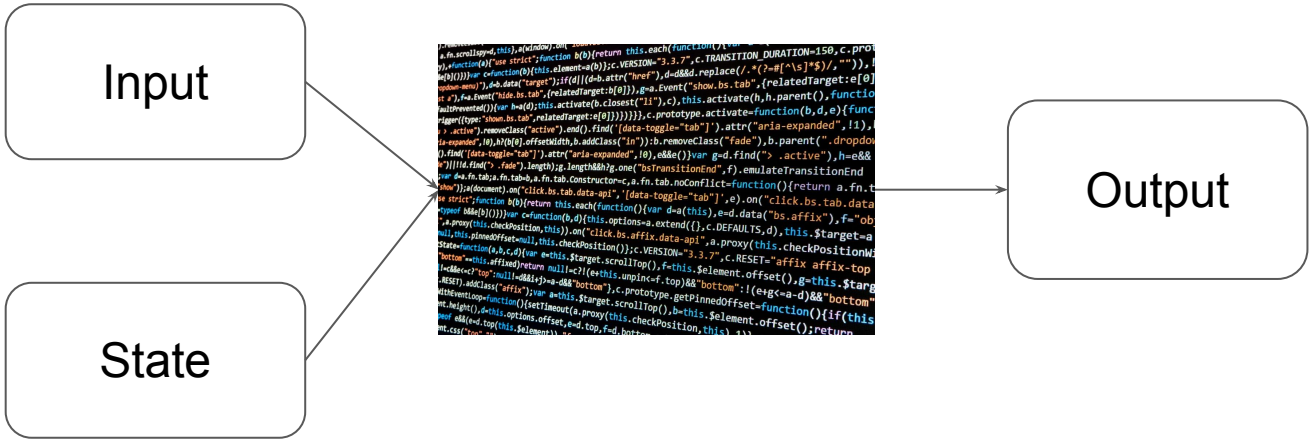
Tools from Certus

- HaRT, Titan, Git-recommend
- https://rubygems.org/gems/git-recommend
- Dipesh test optimization

Testing ML systems?

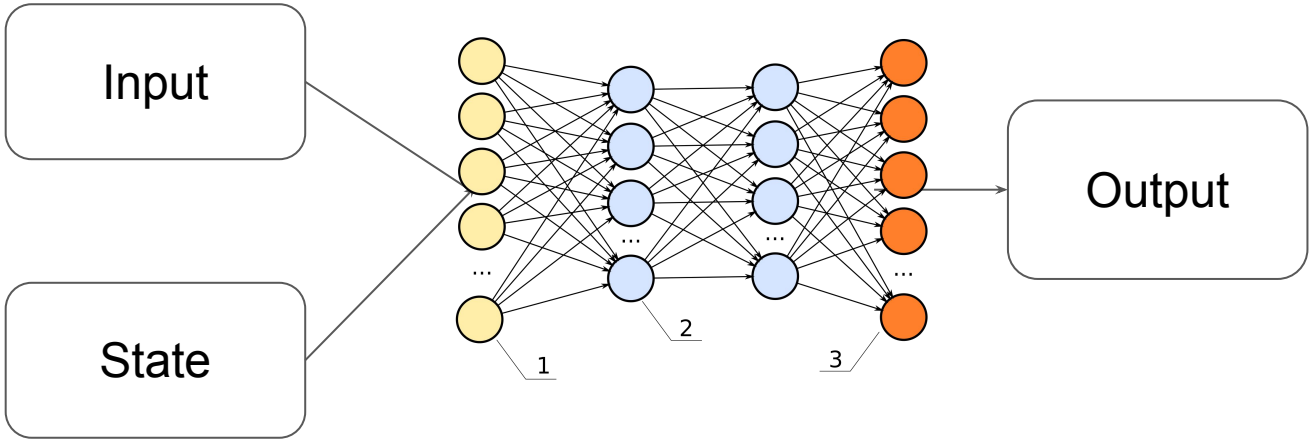

Input
State
Program
Output

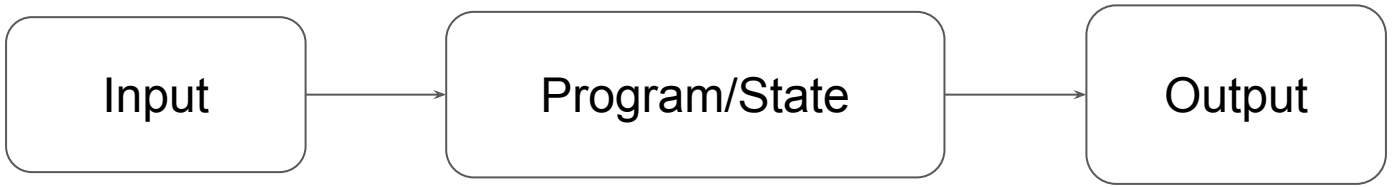
Given *Input* and *State*, is *Output* as expected?



This is traditional testing.

???


Input
State
Output
1
2
3



Need to reason about
Possibly millions of examples!

Many valid outputs...

Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley,gholt,dgg,edavydov}@google.com
{toddphillips,ebner,vchaudhary,mwyoung}@google.com
Google, Inc

Abstract

Machine learning offers a fantastically powerful toolkit for building complex systems quickly. This paper argues that it is dangerous to think of these quick wins as coming for free. Using the framework of *technical debt*, we note that it is remarkably easy to incur massive ongoing maintenance costs at the system level when applying machine learning. The goal of this paper is highlight several machine learning specific risk factors and design patterns to be avoided or refactored where possible. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, changes in the external world, and a variety of system-level anti-patterns.

Testing machine learning systems

Harder than ever to reason about the program and its state

Validating code -> validating input and output

Thinking in code -> thinking in statistical models

The future?

More efficient testing and debugging

Bots will do the boring work

Humans will be guides an sanity checkers

Testing will be more data-driven and data-oriented

Testing Machine Learning systems will be a big challenge

Thank you!