Google

All of Our ML Ideas Are Bad

(and We Should Feel Bad)

SRECON EMEA 2019

Oct, 2019

tmu@google.com

Google

All of Our ML Ideas Are Bad

(and We Should Feel Bad)

SRECON EMEA 2019

Oct, 2019

@tmu*

tmu@google.com

* But mostly only for Yo support

Quick Intro

Agenda

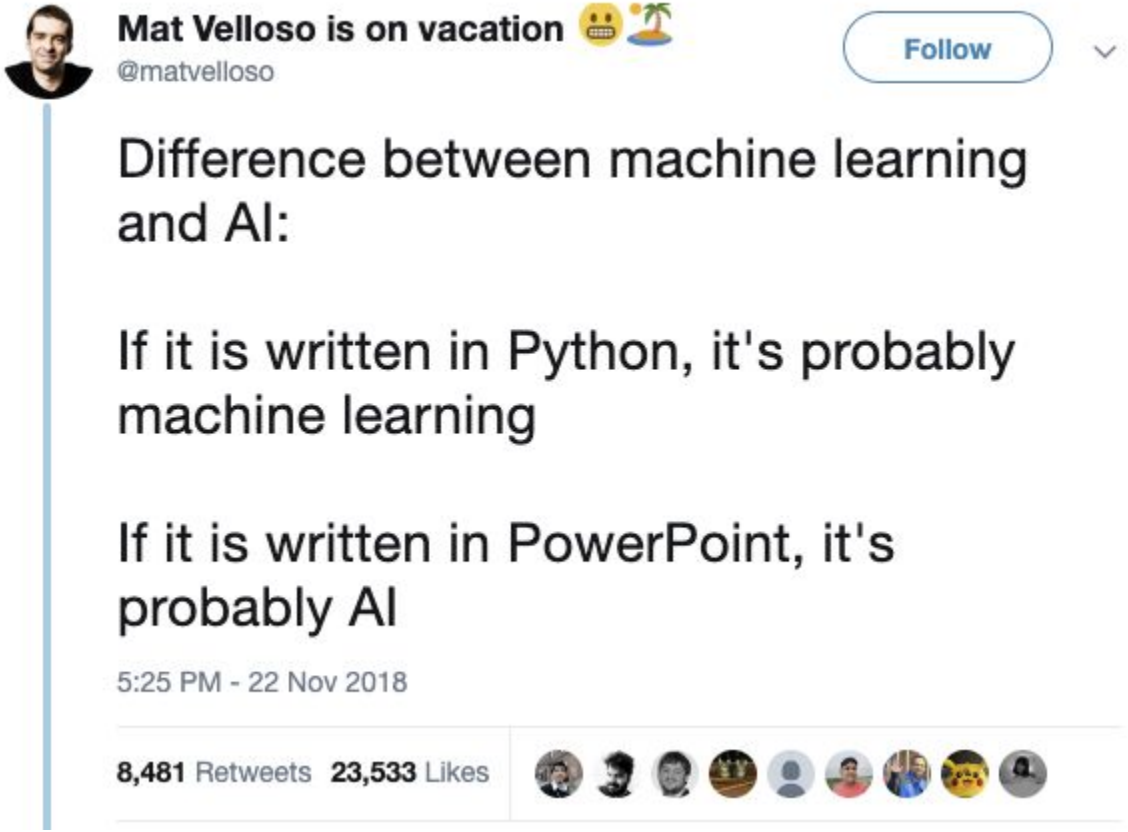
- Basic Questions/Intro (including philosophy, humor and personal info)
- A (hopefully short) (extremely basic) Machine Learning Primer
- ML for Production: Ideas and why they are (mostly) bad.
- Are there any **Good** production uses for ML?
- Future work (for someone)

Basic Questions

- **Who am I?**
  - Recovering Systems/Network engineer, 10+ years working on large ML at Google, Lead ML for SRE, based in Pittsburgh, PA, US. There's other facts. But those are some of them.
- **Who are you?**
  - SREs of all stripes interested in making production work. Not data scientists. Have not used ML in anger on purpose.
- **Why are we here?**
  - Profound philosophical answers. But today, mostly make our lives better.

- **ML vs AI: What is actually happening here?**

ML vs. AI


Mat Velloso is on vacation
@matvelloso
Follow
Difference between machine learning and AI:
If it is written in Python, it's probably machine learning
If it is written in PowerPoint, it's probably AI
5:25 PM - 22 Nov 2018
8,481 Retweets 23,533 Likes

ML vs AI

AI

A family of technologies and approaches designed to create machines that can demonstrate “intelligence” that we normally associate with humans. Ideally this would include something like “learning” and “problem solving”.

ML

The study of algorithms that enable computer systems to solve some specific problem/perform some task by learning from data. A subset of AI.

Mostly care about this today.

Doesn’t Really Matter

People use these terms inconsistently.

Language is about communication and carefully parsing “AI” vs “ML” is a terrible way to communicate. We should learn not to do that.

Machine Learning:
A (very, very) Short Primer

Apples vs. Oranges.

The problem we're solving is: take some data (let's say an image or a preprocessed image) and apply the (hopefully correct!) label of "Apple" or "Orange" to it.

How would we do this in software?

Apples

Apples are red.

https://www.pexels.com/photo/apple-fruit-healthy-food-39803/

Oranges

Oranges are orange.

Cool. Done?

https://www.pexels.com/photo/food-healthy-orange-white-42059/

Green Apples

So some apples are green.

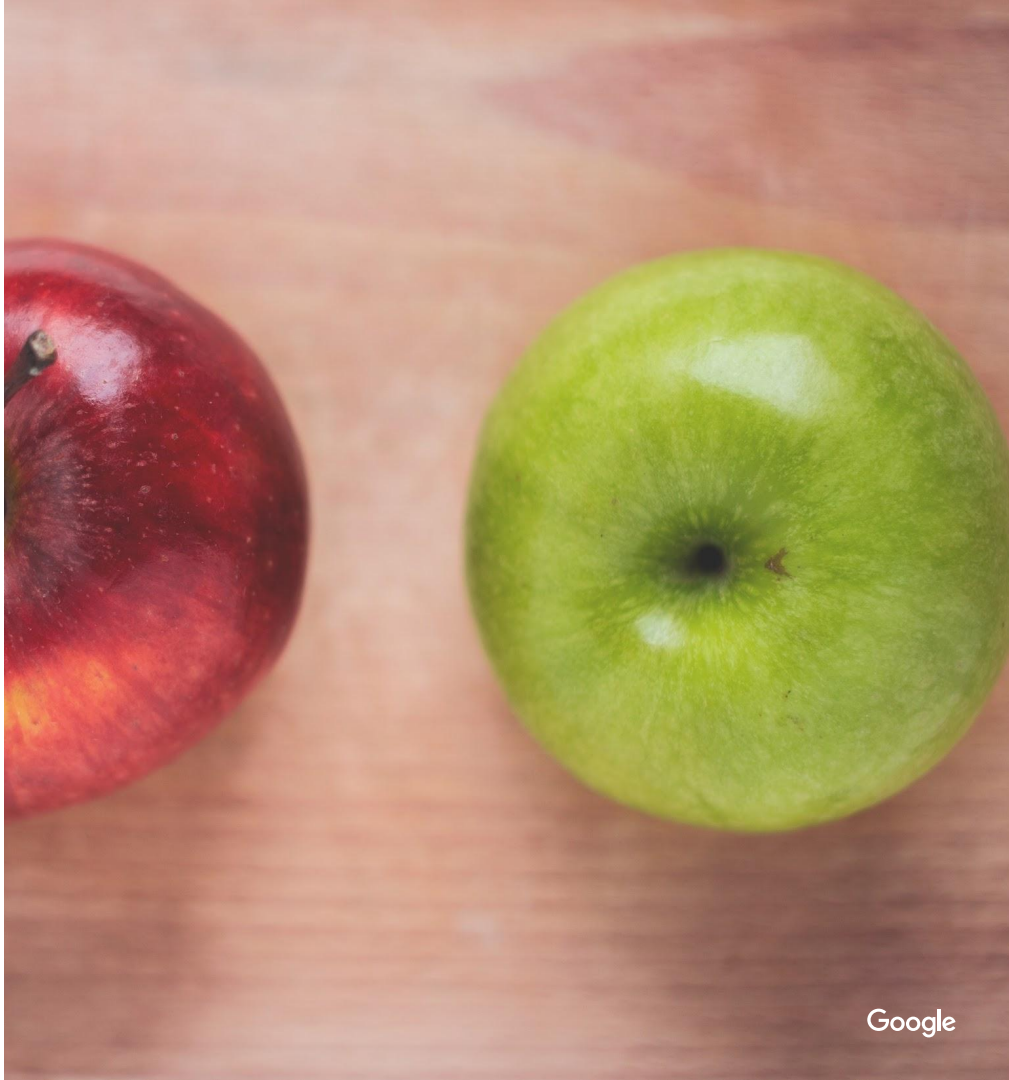
OK, fine.

So apples are Red or green.

Oranges are orange.

Done!

https://www.pexels.com/photo/apples-confection-crispy-delicious-135130/



A Green Orange?

https://www.pexels.com/photo/pile-of-orange-fruits-2495220/

And Yellow?

OK.

Apples are Red or Green
Oranges are Orange or Green or Yellow (or a blend of two of those).

https://www.pexels.com/photo/photo-of-pile-of-oranges-2288688/

Or a Blend

OK.

Apples are Red or Green

Oranges are Orange or Green or Yellow (or a blend of two of those).

Oranges are bumpy and Apples are smooth!

https://www.pexels.com/photo/photo-of-orange-fruits-in-trays-1683916/

Bumpy Apple?

A wet apple might **look** bumpy to our image processing.

https://www.pexels.com/photo/close-up-of-fruits-hanging-on-tree-257840/

Bumpy Apple?

A multichromatic apple might look bumpy, too.

https://www.pexels.com/photo/close-up-photography-of-honeycrisp-apples-and-two-clear-glass-bottles-filled-with-liquid-1532193/

A Green, Bumpy Apple?

https://www.pexels.com/photo/apple-color-delicious-diet-1161467/

Some Oranges are Smooth

Or look that way.

https://www.pexels.com/photo/mandarin-fruit-2135677/

It Gets Harder

https://www.pexels.com/photo/orange-fruit-36775/

Now What?

https://www.pexels.com/photo/citrus-fruits-slice-793763/

A Better Approach: Learn From Examples

What we need:

A bunch of **labeled examples.**

A system for updating some kind of data structure from some fact about those examples. Facts about examples are called **features**. What we know about them might be stored in a **map** or a **graph**

A **cost function** to evaluate how well we're doing.

Lots of different ways to do this and for purposes of this talk it doesn't matter.

ML for Production

What (SRE Thing) is ML **For?**

We want ML to do the annoying/repetitive/boring (but important) stuff:

- Monitoring maintenance - updating thresholds and parameters
- Alert suppression/analysis - flagging false positives, suppressing them
- Capacity Planning/Prediction
- Validation of Canaries - stopping bad pushes
- Root Cause Analysis of Incidents/Outages
- Automatic Service/Resource Scaling

What's So Wrong With That?

That sounds lovely.

Seriously.

Why **shouldn't** we want those things to be handled by the computers?

And why **can't** we have that?

Examples….

Categorizing/Prioritizing Tickets

Too many tickets?

Just tell us which ones really matter!

Photo by Daniel Bradley on Unsplash

Tickets: The Idea

In an attempt to focus and prioritize work, you create tickets in a single stream for your team

Humans and computers together create too many tickets

Spend all of team's time triaging and categorizing tickets.

Use Machine Magic!!!
...
...
... and profit?

Ticket Handler Algorithm

Step 1

- Determine the best features for predicting ticket priority either by hand or using an algorithm
- (It's algorithms all the way down)

Step 2

- Get labels from existing tickets by looking at some signal from human processing.
- Final priority == priority label
- Closed (Will Not Fix, Obsolete, Working as Intended) is negative signal
- Closed (Fixed) and perhaps tagged to a post mortem is a positive signal

Step 3

- Train a supervised learning model on these labeled example tickets using the above features

Step 4

- Use the model to set the priority for inbound tickets.

Step 5

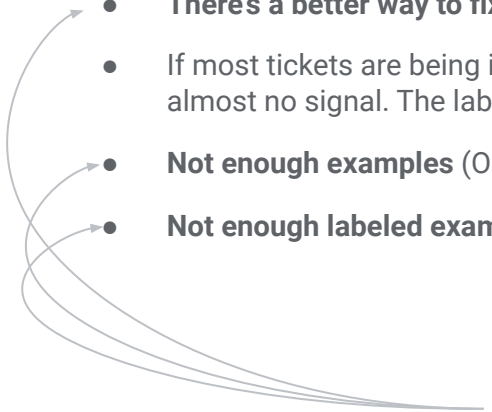
- Ticket bliss. All tickets are now the right priority at ingress.

A Machine Learning Algorithm Walks Into a Bar

A Machine Learning Algorithm Walks Into a Bar

What, if anything, is wrong with this?

- **There's a better way to fix** this problem (stop false-positive tickets!)
- If most tickets are being ignored or going un-worked now, there's almost no signal. The labels that we have may be quite bad.
- **Not enough examples** (O(10K) or O(100K) or so?)
- **Not enough labeled examples** (O(100) or O(1k) or so?)



Remember these

Automatic Root Cause Analysis

Given an (incident,outage,page) tell us what caused it.

Photo by James Wheeler from Pexels

RCA: The Idea

Outages are difficult to troubleshoot because so many things can contribute. The best production engineers are way faster than the least experienced.

Outages last too long and new staff take too long to train up on complex systems.

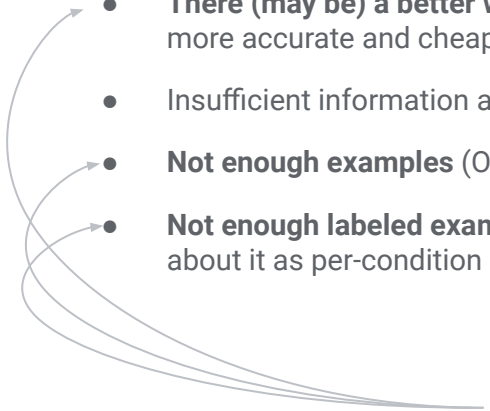
Given an (alert, outage, incident) production engineering staff need the most probably relevant information surfaced.

Use Machine Magic!!!
…
…
… and profit?

Great!

What, if anything, is wrong with this?

- **There (may be) a better way to fix** this problem: Heuristics may be more accurate and cheaper, possibly with high false positives.
- Insufficient information about the accuracy of what we already do.
- **Not enough examples** (O(10K)?)
- **Not enough labeled examples** (O(1k) per year, but really if you think about it as per-condition or per-cause it's O(100) or less).



Our friends are back.

ML Canarying

Automatically detect bad pushes of software under a variety of circumstances

Photo by Dick Donaghue from Pexels

Canarying: The Idea

We have software that we change. We subscribed to CICD religion and now want to push this software to production, all the time.

Somehow our software is not perfect all of the time. Sometimes it's even badly broken.

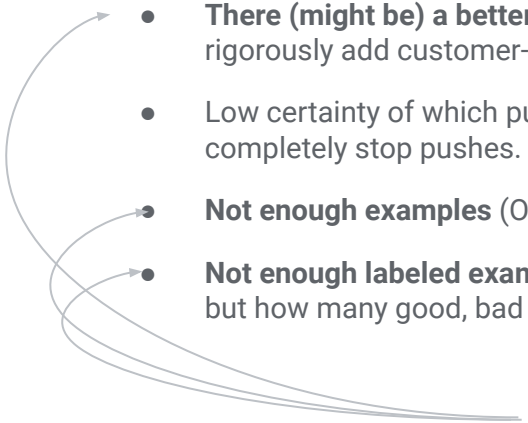
We write tests (smoketests, integration tests) but we keep missing stuff. And sometimes our monitoring doesn't catch all of these cases so our users suffer.

Use Machine Magic!!!
...
...
... and profit?

Automated Canary Analysis

What, if anything, is wrong with this?

- **There (might be) a better way to fix** this problem (add tests and rigorously add customer-emulating, SLO-based tests after outages).
- Low certainty of which pushes are actually bad. False positives can completely stop pushes.
- **Not enough examples** (O(100) pushes per year, maybe O(1k))
- **Not enough labeled examples** (O(10) or O(100) bad pushes per year but how many good, bad pushes are labeled with confidence?).



This is getting tedious; no one likes a know-it-all

Most of what people call ML is Marketing ML

Solve the underlying problem at the source.

Use the least amount of math necessary. Tell your management it's AI/ML if that works.

Save fancy math for very, very big problems.

Plausible (Production) Uses of ML

ML isn't **useless** *(but finding good uses is hard)*

Requirements for Plausible ML Production Applications

A Problem	• An ML-shaped problem: something needs to be **clustered** or **categorized** or **predicted**.
Data	• Loads of data. Tons. Tonnes. Scads. Seriously a lot of data. You just won't believe how vastly, hugely, mindbogglingly much data you'll need. I mean, you may think it's a long way down the road to the chemist, but that's just peanuts compared to the data you'll need. • Think millions to hundreds of millions of data points. Not thousands. If it's thousands, just look at them by hand or with traditional analytics.
Labeled Data	• Need thousands to millions of examples that we understand/know. • Need high confidence (low error rate and understood error bias) in these labels.
Features	• Some ideas about what **part** of our data is most predictive/valuable.
Objective Function	• Some clever way of evaluating whether we did something useful with our model.

Most Important Pre-Work

Gather your data.

Seriously, it’s all about the data.

Data.

Data.

….

….

…. Data ...

Do Now:
Recommendations for Production ML Preparation

Think

Consult with people who know your applications, infrastructure, customers. Brainstorm ideas for useful ways to think about how to understand and manage (and automate) these.

Gather

Put your data in a place where you can find it, extract it, combine it, and process it. You won't be able to do anything until your data are sorted. So sort them.

Learn

Do some basic ML training/thinking. In particular, in addition to basic concepts, also learn the **platforms** that are out there (TFX, Amazon Sagemaker, Google Cloud ML Engine, Microsoft Machine Learning Service). Be ready to prototype ideas once you have them on data once you've gathered it.

Future Work

Future Work
(For someone more clever and more ambitious)

Ontology

What there is. Someone needs to make curated more careful categorization of the problems we think are amenable to using ML. It would be useful to think about what **else** those problems have in common other than “we think the computers should magically do this for themselves.”

Epistemology

What (and how) we know. We need to know what we know about these problems and our infrastructure in general. What are the data and relationships that we have and could have.

This should map onto requirements of implementation, of course.

Metaphysics

Abstract theory with no basis in reality (i.e. AI). :-)

Target a few implementation ideas for ML control/troubleshooting and iterate.

Thank You

Photo by Pixabay from Pexels

Google