Verification and Validation of AI Software

Peter Norvig

Google
“How can we reinvent it knowing that software can play such an important role.”
“How can we reinvent it knowing that artificial intelligence can play such an important role.”
New Classes of Applications
Computer Science:
Doing the right thing, efficiently, when you can define what that means

Artificial Intelligence:
Doing the right thing, efficiently, when you don’t know what to do
Visual Object Recognition
Traditional Software vs Machine Learning
Traditional Software

Decision 1:
- True: Sequence 1
- False: Decision 2

Decision 2:
- True: Sequence 2
- False: Decision 3

Decision 3:
- True: Sequence 3
- False: Sequence 4
Machine Learning
Example: Spelling Correction
My colleague, Mehran Sahami, worked on machine learning and computer vision.
if (is_before('i', 'e') and not is_after('i', 'c')):
    return CORRECT

2000+ lines of code per language
Résultats pour Mehran *sahami*
Essayez avec l'orthographe Mehran Salami

Images correspondant à mehran sahami

Plus d'images pour mehran sahami
21 lines of code total

\[ \text{return } \text{argmax}(P(\text{observed}|w) \times P(w) \text{ for } w \text{ in dictionary}) \]

def \text{train}(\text{features}):
    \text{model} = \text{collections.defaultdict}(\lambda: 1)
    for f in \text{features}:
        \text{model}[f] += 1
    \text{return model}

def \text{words}(\text{text}):
    \text{return } \text{re.findall('[a-z]+', text.lower())}

\text{P} = \text{train}(\text{words(file('big.txt').read())})

\text{alphabet} = 'abcdefghijklmnopqrstuvwxyz'

def \text{edits1}(\text{word}):
    \text{splits} = [(\text{word}[i], \text{word}[i:]) for i in range(len(\text{word}) + 1)]
    \text{deletes} = [a + b[1:] for a, b in \text{splits} if b]
    \text{transposes} = [a + b[1] + b[0] + b[2:] for a, b in \text{splits} if len(b)>1]
    \text{replaces} = [a + c + b[1:] for a, b in \text{splits} for c in \text{alphabet} if b]
    \text{inserts} = [a + c + b for a, b in \text{splits} for c in \text{alphabet}]
    \text{return set(\text{deletes} + \text{transposes} + \text{replaces} + \text{inserts})}

def \text{known_edits2}(\text{word}):
    \text{return set(\text{e2} for \text{e1} in \text{edits1}(\text{word}) for \text{e2} in \text{edits1}(\text{e1}) if \text{e2} in \text{P})}

def \text{known}(\text{words}):
    \text{return set(\text{w} for \text{w} in \text{words} if \text{w} in \text{P})}

def \text{correct}(\text{word}):
    \text{candidates} = (\text{known}([\text{word}]) or \text{known}(\text{edits1}(\text{word}))) or
                      \text{known_edits2}(\text{word}) or [\text{word}]
    \text{return max(\text{candidates}, key=\text{P}.get)
Object Recognition via Supervised Machine Learning
Object Clustering via Unsupervised Machine Learning
Inventory
Object Recognition via Unsupervised Machine Learning

Inventory
Choose a set of, say, 1000 Pieces to make near-copies of each Image minimizing difference:

\[
\sum \left( \text{Copy}_{x, y} - \text{Image}_{x, y} \right)
\]

where

\[
\text{Copy}_{x, y} = \sum \text{weight}_i \times \text{Piece}_i
\]
Inventory?
10,000,000 YouTube video frames
**Synset:** rust, rust fungus
**Definition:** any of various fungi causing rust disease in plants.
*Popularity percentile:* 69%
*Depth in WordNet:* 6

**Synset:** fungus
**Definition:** an organism of the kingdom Fungi lacking chlorophyll and feeding on organic unicellular or multicellular organisms to spore-bearing syncytia.
*Popularity percentile:* 60%
*Depth in WordNet:* 5

**Synset:** honey mushroom, honey fungus, Armillariella mellea
**Definition:** a honey-colored edible mushroom commonly associated with the roots of trees; do not eat raw.
*Popularity percentile:* 56%
*Depth in WordNet:* 8

**Synset:** white fungus, Saprolegnia ferax
**Definition:** a fungus that attacks living fish and tadpoles and spawn causing white fungus hyphae on especially peripheral parts (as fins).
*Popularity percentile:* 53%
*Depth in WordNet:* 6

**Synset:** sac fungus
**Definition:** any of various ascomycetous fungi in which the spores are formed in a sac.
*Popularity percentile:* 49%
*Depth in WordNet:* 6
ImageNet
2012 Model
8 layers
U Toronto Team: Krizhevsky, Sutskever & Hinton

16.4% top-5 error rate
ImageNet

2014 Model

24 layers

Google Team

6.6% top-5 error rate
End-to-End Caption Writing

Vision
Deep CNN

Language Generating
RNN

A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.
Human: Three different types of pizza on top of a stove.
Machine: Two pizzas sitting on top of a stove.
A couple of giraffes standing next to each other
A reflection of a dog in a side view mirror
A man riding a skateboard
Challenges for Machine Learning Systems
EXPLAINING AND HARMONIZING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow, shlens, szegedy}@google.com

ABSTRACT

Several machine learning models, including neural networks, consistently misclassify adversarial examples—inputs formed by applying small but intentionally worst-case perturbations to examples from the dataset, such that the perturbed input results in the model outputting an incorrect answer with high confidence. Early attempts at explaining this phenomenon focused on nonlinearity and overfitting. We argue instead that the primary cause of neural networks’ vulnerability to adversarial perturbation is their linear nature. This explanation is supported by new quantitative results while giving the first explanation of the most intriguing fact about them: their generalization across architectures and training sets. Moreover, this view yields a simple and fast method of generating adversarial examples. Using this approach to provide examples for adversarial training, we reduce the test set error of a maxout network on the MNIST dataset.
$x$

“panda”
57.7% confidence

$+.007 \times \text{sign}(\nabla_x J(\theta, x, y))$

“nematode”
8.2% confidence
$x$

“panda”
57.7% confidence

$+ .007 \times \text{sign}(\nabla_x J(\theta, x, y))$

“nematode”
8.2% confidence

$\varepsilon \text{sign}(\nabla_x J(\theta, x, y))$

“gibbon”
99.3% confidence

$= x + \varepsilon \text{sign}(\nabla_x J(\theta, x, y))$
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.
Lack of Clear Abstraction Barriers
Learning to Divide and Conquer: Applying the L* Algorithm to Automate Assume-Guarantee Reasoning

Corina S. Păsăreanu
Perot Systems, NASA Ames Research Center, N269-230, Moffett Field, CA 94035, USA

Dimitra Giannakopoulou
RIACS, NASA Ames Research Center, N269-230, Moffett Field, CA 94035, USA

Mihaela Gheorghiu Bobaru
Department of Computer Science, University of Toronto, 10 King’s College Road, Toronto, Ontario, CANADA M5S 3G4

Jamieson M. Cobleigh
1
Department of Computer Science, University of Massachusetts, 140 Governor’s Drive, Amherst, MA 01003, USA

Howard Barringer
School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK
Lifecycle Verification of the NASA Ames K9 Rover Executive

Dimitra Giannakopoulou\textsuperscript{1,3} Corina S. Pasareanu\textsuperscript{2,3} Michael Lowry\textsuperscript{3} and Rich Washington\textsuperscript{4}

(1) USRA/RIACS \\
(2) Kestrel Technology LLC \\
(3) NASA Ames Research Center, Moffett Field, CA 94035-1000, USA \\
{dimitra, pcorina, lowry}@email.arc.nasa.gov \\
(4) Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA \\
rwashington@gmail.com

Abstract

Autonomy software enables complex, robust behavior in reaction to external stimuli without human intervention. It is typically based on planning and execution technology. Extensive verification is a pre-requisite for autonomy technology to be adopted in high-risk domains. This verification is challenging precisely because of the multitude of behaviors enabled by autonomy technology.

This paper describes the application of advanced verification techniques for the analysis of the Executive subsystem of the NASA Ames K9 Rover. Existing verification tools were extended in order to handle a system the size of the Executive. A divide and conquer approach was critical for scaling. Moreover, verification was performed in close collaboration with the system developers, and was applied during both design and implementation. Our study demonstrates that advanced verification techniques are crucial for real-world planning and execution systems. Moreover, it shows that when verification proceeds hand-in-hand with software development throughout the lifecycle, it can greatly improve the design decisions and the quality of the resulting plan execution system.

and effort since they may involve major changes in the architecture of the system, and possible re-implementation of a large part of it. Therefore, we believe that the verification of a safety critical system should be addressed \textit{as early as possible during its design}, and should go hand-in-hand with later phases of software development.

![Figure 1. Compositional verification throughout the software lifecycle](image.png)

Our work advocates the use of a combination of formal analysis techniques and testing to analyze autonomous
Non-Modularity: Changing Anything Changes Everything
Google Now

**Depart now for:**
Return Rental Car to Logan Airport
156 Tomahawk Dr,
Boston MA

**Time of travel:**
23 minutes by bicycle

```python
event = ExtractEvent(email.body)
trip = Travel(current.location, event.location, event.time)
CreateAlert(trip)
```
Mars Polar Lander

Guidance system initialization (L - 15 min)
4,800 km
5,700 m/s

Turn to entry attitude (L - 12 min)
3,000 km
5,800 m/s

Cruise ring separation/microprobe separation (L - 10 min)
2,300 km
6,200 m/s

Atmospheric entry (L - 5 min)
125 km
6,800 m/s

Parachute deployment (L - 2 min)
8,800 m
460 m/s

Heatshield jettison (L - 110 sec)
7,500 m
250 m/s

Radar ground acquisition (altitude mode) (L - 50 sec)
2,500 m
85 m/s

Radar ground acquisition (Doppler/speed and direction mode) (L - 36 sec)
1,400 m
80 m/s

Lander separation/powered descent (L - 35 sec)
1,300 m
80 m/s

Entry, descent and landing

Solar panel/instrument deployments (L + 20)
Feedback Loops
Attractive Nuisance

“Synonyms”
Privacy and Security
Lack of Intuition
Lack of Tooling
Data/Config Dependencies
Build / Test / Release Cycles
Fundamental Formula of AI

\[ \text{act}^* = \arg\max_{a \in \text{actions}} E(\text{Utility}(a, s)) \]
Concrete Problems in AI Safety

Dario Amodei*
Google Brain

Chris Olah*
Google Brain

Jacob Steinhartd
Stanford University

Paul Christiano
UC Berkeley

John Schulman
OpenAI

Dan Mané
Google Brain

Abstract

Rapid progress in machine learning and artificial intelligence (AI) has brought increasing attention to the potential impacts of AI technologies on society. In this paper we discuss one such potential impact: the problem of accidents in machine learning systems, defined as unintended and harmful behavior that may emerge from poor design of real-world AI systems. We present a list of five practical research problems related to accident risk, categorized according to whether the problem originates from having the wrong objective function ("avoiding side effects" and "avoiding reward hacking"), an objective function that is too expensive to evaluate frequently ("scalable supervision"), or undesirable behavior during the learning process ("safe exploration" and "distributional shift"). We review previous work in these areas as well as suggesting research directions with a focus on relevance to cutting-edge AI systems. Finally, we consider the high-level question of how to think most productively about the safety of forward-looking applications of AI.
Avoiding Negative Side Effects
Avoiding Reward Hacking
Mechanism Design
Safe Exploration
Inattention Valley
Transfer Learning
Scalable Oversight
“the worst ... except all the others that have been tried”