大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

AIのテストにおける課題と技術
そこから考える今後のテスト

国立情報学研究所　石川　冬樹
f-ishikawa@nii.ac.jp / @fyufyu
http://research.nii.ac.jp/~f-ishikawa/

自己紹介

- 国立情報学研究所　准教授
  - ソフトウェア工学および，先端自律・スマートシステムに関する研究・教育
- 電気通信大学　客員准教授
  - 社会人博士学生の指導（在学中も含め計9名）
- 産業界向け教育・実践研究
  - トップエスイー（形式手法，クラウド等）
  - 日科技連SQiP研究会（要求と仕様，次年度はAI＋品質）
  - 電通大社会人向けコース（WebSys・AISEC）

興味：要求や仕様、設計に関する様々なモデルを活用した検証・推論・最適化・自動テスト生成・自己適応など

研究者としての最近の主な活動

- 自動（運転）車のテスト・検証
  - 物理世界や確率などの連続値や微分方程式を含み複雑なシステムのテスト・検証・品質保証
  - 形式検証・テスト自動生成・最適化・機械学習などの技術を活用・併用する融合アプローチの追求
- 機械学習（や広くAI）を含むシステムの品質保証
  - コミュニティ活動，概念や技術の整理
  - テスト・検証技術の研究
- 形式手法の活用支援
  - 形式仕様の段階的詳細化における保守や再利用
  - 様々な手法・ツールの活用支援

MLSE研究会 メルシー

https://sites.google.com/view/sig-mlse/

- 日本ソフトウェア科学会 機械学習工学研究会
  - 2018年4月 正式立ち上げ
  - 研究会なので，基本は「場」の提供に専念
- 2年経たずに30回近くのイベント
  - 他イベント内のセッションも多い（分野横断的）
  - 参加者は企業の方々が中心
    （学界からの講演会や国際会議の輪講でも）
  - 焦点を絞ったいくつかのWGが進行中
- まだまだこれから
  - 学界からの貢献，異なる分野からの技術の連携・融合，現場経験に基づいた取り組み強化が必要

QA4AIコンソーシアム

http://www.qa4ai.jp/

- QA4AI： AIプロダクト品質保証コンソーシアム
  - これも2018年4月発足
  - AIプロダクトの品質保証に関する調査・体系化，適用支援・応用，研究開発，社会の啓発を推進
  - 現在 39名 + 3団体

参加＆貢献大歓迎

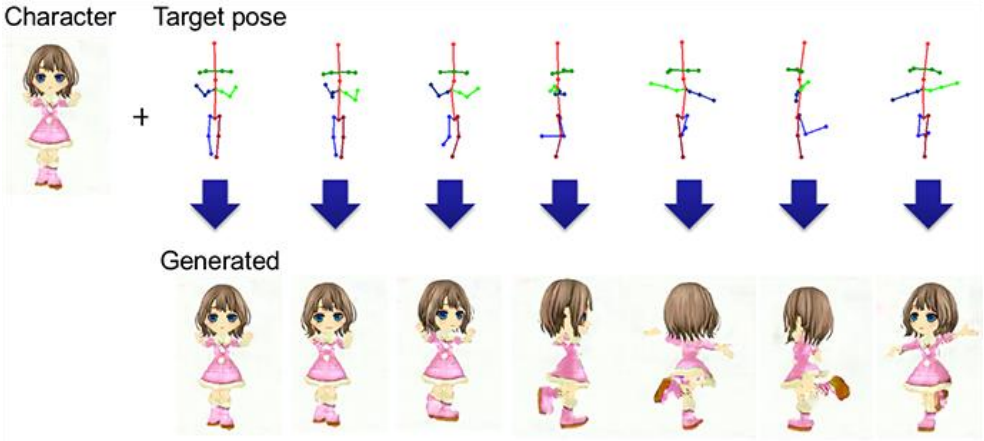
149ページ！

- 2019年5月17日ガイドライン初版リリース
  ＋ 5団体での合同シンポ（今後連携！）
  - 自動運転の章もあり
- 他の取り組みと連携しつつ継続更新中

「AIの品質」を考える事例

技術の進化（ごくごく一例）

- DeNAさん（2018年5月）
  - 画像に対し指定した姿勢の画像を生成可能



画像元：[https://dena.com/intl/anime-generation/images/anime2.png]

  - 動画を生成することもでき，「始点」から「終点」へ連続的に「変身」させることも（服を変えるなど）

ところで皆さんなら何をどう保証（テスト）しますか？

[https://dena.com/intl/anime-generation/]

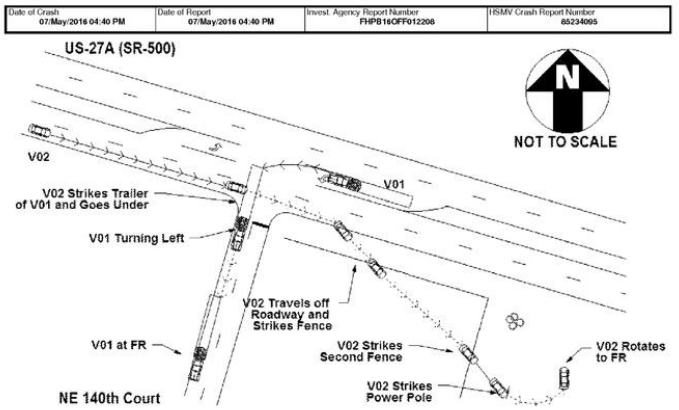
社会的影響の大きい例・使い方を問う例

- 2016年のテスラ自動運転車の事故　*ここが機械学習*
  - テスラの発表：「まぶしい空に対してトレーラーの白い側面を認識できず」（「運転者すら」ともある）

[https://www.tesla.com/jp/blog/tragic-loss]

  - 2017年11月の調査報告での論点に上記は全くなく，「そもそも自動運転の対象外状況だが運転者が長らく操作していない」という過信や警告音の効力が中心

[https://dms.ntsb.gov/pubdms/search/hitlist.cfm?docketID=59989]



[http://www.dailymail.co.uk/news/article-3677101/Tesla-told-regulators-fatal-Autopilot-crash-nine-days-happened.html]

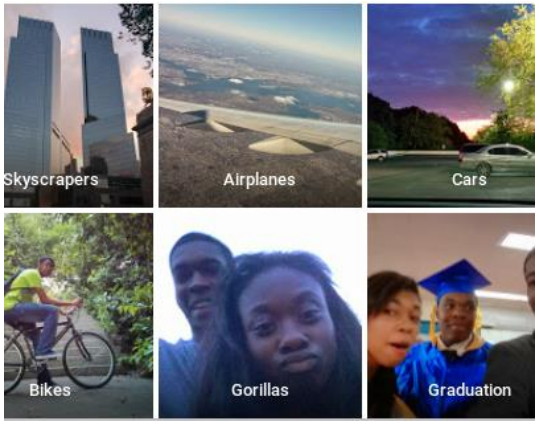
社会的影響の大きい例・全体設計を問う例

- 2018年のウーバー自動運転車の事故
  - 車道上での歩行者死亡事故
  - 当初はドライバーが集中していなかったということが焦点だったが，調査の後には設計が焦点に
  - 車道上にあるものは「自動車」「自転車」「その他」にのみ分類する設計
  - 自動運転時は緊急ブレーキオフ（運転手判断を優先するため）
  - 衝突直前には，時刻により違う物体として識別することを繰り返していた

[https://dms.ntsb.gov/pubdms/search/hitlist.cfm?docketID=62978]

社会的影響の大きな例・技術的限界の例

- Googleフォトの画像認識
  - 被写体の自動タグ付け機能
  - 黒人を「ゴリラ」とタグ付け
  - 2年経って本質的には直せていない
    （ゴリラを禁止ワード扱い
    にして対策）

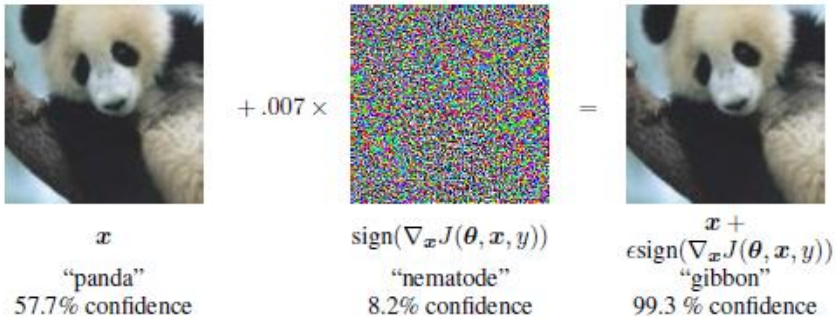


[https://www.theguardian.com/technology/2015/jul/01/google-sorry-racist-auto-tag-photo-app]

[https://www.theguardian.com/technology/2018/jan/12/google-racism-ban-gorilla-black-people]

よく知られた課題（の一つ）：敵対的サンプル

- 優れた画像識別器が少しのノイズで誤認識

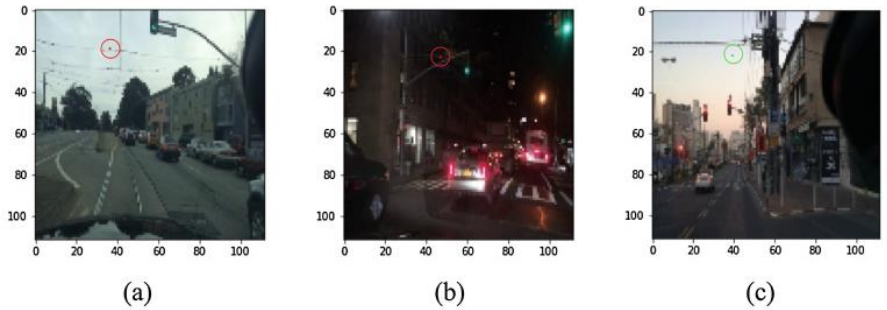


有名な例
「パンダ」が「テナガザル」に

[Goodfellow et al., Explaining and Harnessing Adversarial Examples, 2015]

物理的なテープ貼付などによる誤認識発生

[Ackerman, Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms, 2017]



1ピクセルの変化で信号色の誤認識発生

[Wicker et al., Feature-Guided Black-Box Safety Testing of Deep Neural Networks, 2018]

訓練データ・ある種の攻撃に関する例

- Twitter Botによる不適切発言
  - 差別や放送禁止用語を「教えた」ユーザがいた

If you guessed, “It will probably become really racist,” you’ve clearly spent time on the Internet. Less than 24 hours after the bot, @TayandYou, went online Wednesday, Microsoft halted posting from the account and deleted several of its most obscene statements.

The bot, developed by Microsoft’s technology and research and Bing teams, got major assistance in being offensive from users who egged it on. It disputed the existence of the Holocaust, referred to women and minorities with unpublishable words and advocated genocide. Several of the tweets were sent after users commanded the bot to repeat their own statements, and the bot dutifully obliged.

[https://www.nytimes.com/2016/03/25/technology/microsoft-created-a-twitter-bot-to-learn-from-users-it-quickly-became-a-racist-jerk.html]


TECHNOLOGY

Microsoft Created a Twitter Bot to Learn From Users. It Quickly Became a Racist Jerk.

By DANIEL VICTOR MARCH 24, 2016

Tay’s Twitter account. The bot was developed by Microsoft’s technology and research and Bing teams.

バイアス・倫理的要求に関する例

- Amazonの「AI採用」が男女差別をしていた？
  - 過去のデータは「教師」として現在でも適切か？
  - 訓練データの偏り
- 2019年12月
  - 3学会での合同声明

    http://ai-elsi.org/archives/898


REUTERS ワールド ビジネス マーケット 外国為替 ビデオ

外国為替 株式市場 外国為替フォーラム 貿易摩擦 北朝鮮 トランプ政権 オピニオン 経済・政策 テクノロ

テクノロジー 2018年10月11日 / 15:30 / 1日前

焦点：アマゾンがＡＩ採用打ち切り、「女性差別」の欠陥露呈で

Jeffrey Dastin 2 分で読む

［サンフランシスコ １０日 ロイター］ - 米アマゾン・ドット・コム(AMZN.O)が期待を込めて進めてきたＡＩ（人工知能）を活用した人材採用システムは、女性を差別するという機械学習面の欠陥が判明し、運用を取りやめる結果になった。

[https://jp.reuters.com/article/amazon-jobs-ai-analysis-idJPKCN1ML0DN]

※ 簡単な計算例（もしデータや処理に差別がなくても…）：

男性90人と女性10人（計100人）に関する推論

男性90%正解，女性10%正解 → 82/100点

男性80%正解，女性80%正解 → 80/100点

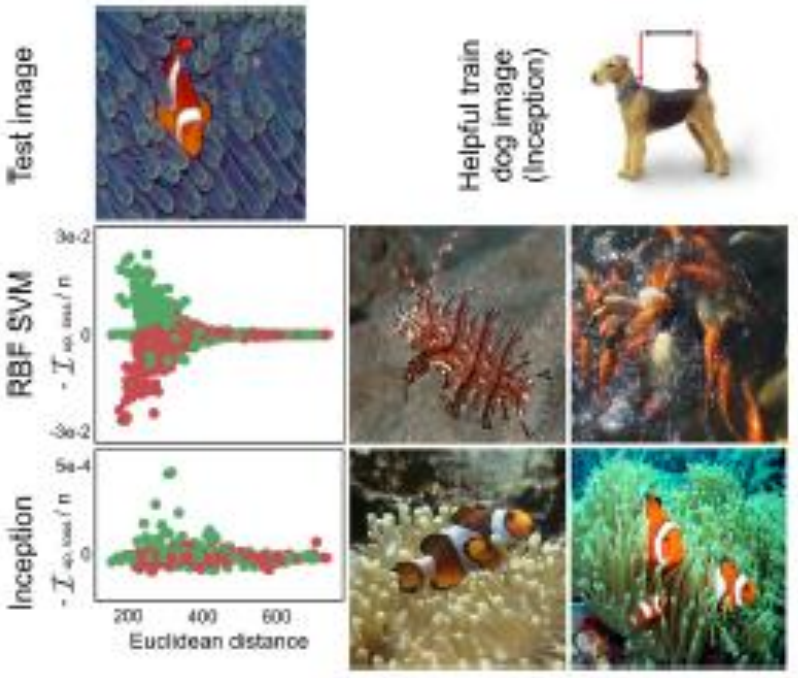
学んだ傾向・規則の妥当性に関する例

- 入力画像のどこが結果に効いたか分析してみた
  - 「雪が写っているときは
    オオカミである」という
    規則を学んでしまった


(a) Husky classified as wolf　(b) Explanation

[Ribeiro et. al., " Why Should I Trust You?":
Explaining the Predictions of Any Classifier, KDD'16]

- どの訓練画像が参考に
  なったか分析してみた
  - 悪い識別器では，「魚」と
    いう結果は合っていても
    雑な色感しかみてない



[Koh et. al., Understanding Black-box Predictions
via Influence Functions, ICML'17]

プロダクト・サービスとしての品質？

- これらすべて「バグ」か？潰さなければならないのか？
- そもそも潰せるものなのか？それとも技術的限界？
- どうやってこれらに気づくのか？他に対処すべき「同種」や「類似」の状況は列挙できるのか？どうやって？
- そもそも何を持って「品質保証」「完成」とするのか？「仕様」は何なのか？事前に見積・合意できるのか？
- 修正内容をどう決めてどれだけの頻度で更新するのか？
- 顧客やユーザには何をどうやって説明するのか？
- こんな挙動の原因を把握したり，予測し踏まえてテストしたりできるのか？どうやって？開発者ならわかるものなのか？外部品質保証者は何ができるのか？
- ・・・

機械学習の本質的な違い

演繹的システム開発と帰納的システム開発

- 演繹的システム開発（従来）
  - 計算や判断を行うための知識・規則（モデル・アルゴリズム）を，人が決めてプログラムという形で書き下す

- 帰納的システム開発（というか機械学習）
  - 計算や判断を行うための知識・規則（モデル・アルゴリズム）を訓練データから獲得し生成する
  - それを行うプログラムを人が書き下す

※広く「AI」というとどちらの作り方もありうるが，
後者の場合，開発・運用・品質保証が大きく変わる

例

この線引きを訓練データから作る

テナガザル

35 24 210	20 121 24	122 81 20
211 54 42	12 222 90	88 79 116
24 36 98	98 181 31	66 31 198

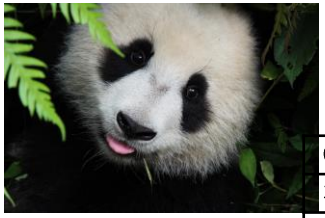
13 83 33	13 45 94	75 74 111
111 8 73	192 1 221	237 31 1
74 35 122	93 76 244	73 211 45

0 245 210	20 12 114	84 99 100
11 86 99	121 88 91	18 0 77
46 87 121	70 76 122	122 14 94

パンダ

254 32 67	222 88 1	108 76 14
12 86 222	98 75 122	111 74 74
198 87 33	188 173 4	68 176 83

77 81 123	122 158 6	76 63 42
3 3 78	19 183 84	76 63 123
98 83 111	123 7 99	253 48 91



0 24 31	20 21 124	12 101 50
21 54 242	112 22 90	8 79 214
124 56 85	98 99 141	166 1 198

[http://free-photos.gatag.net/]

別の言葉： Software 2.0

- 成果物は，大量のパラメータ値
  - 「ソフトウェアの書き方（作り方）」が変わっている

Medium

Andrej Karpathy Follow

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

Nov 11, 2017 · 8 min read

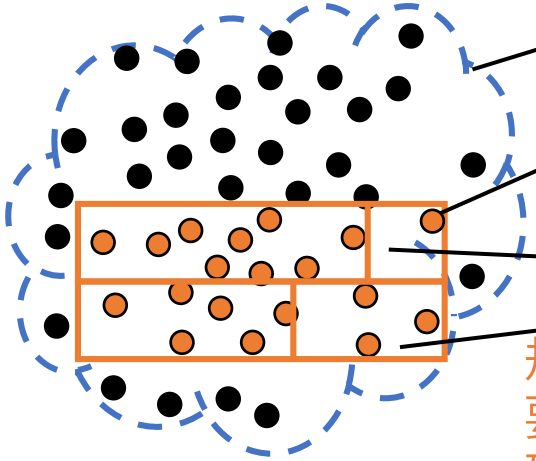
Software 2.0

I sometimes see people refer to neural networks as just “another tool in your machine learning toolbox”. They have some pros and cons, they work here or there, and sometimes you can use them to win Kaggle competitions. Unfortunately, this interpretation completely misses the forest for the trees. Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.

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

従来ソフトウェアと機械学習

■従来ソフトウェア



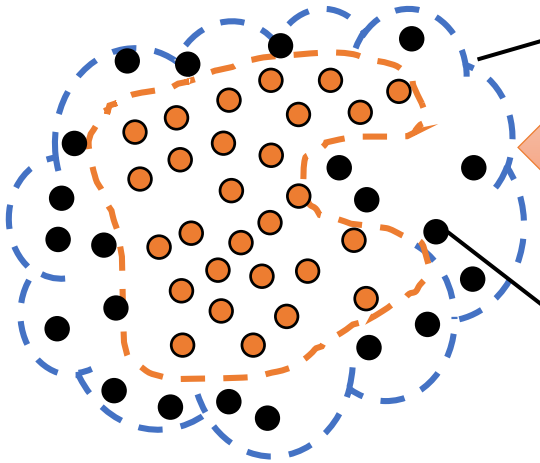
ゴール：名医のように健康状況を適切に判定する

個別事例：Aさんの健康診断データは・・・
→「糖尿病疑い，他問題なし」と判定すべき

状況に応じルール化された要求仕様：
- HbA1c値がｘ以上の場合…
- 面積を基にして…というアルゴリズムで
  レントゲンの影の有無を判定し…

規定された
要求範囲だけ
確実に充足

■機械学習型AI



ゴール：名医のように健康状況を適切に判定する

正解率 90%

ゴール全体を
大まかに達成

データ

過去の
判定結果

データを基に訓練して実現した機能

個別事例：Aさん（自明な例でも誤った判定が起きうる）

高度な認識や判断が可能になる一方で思わぬ誤りも

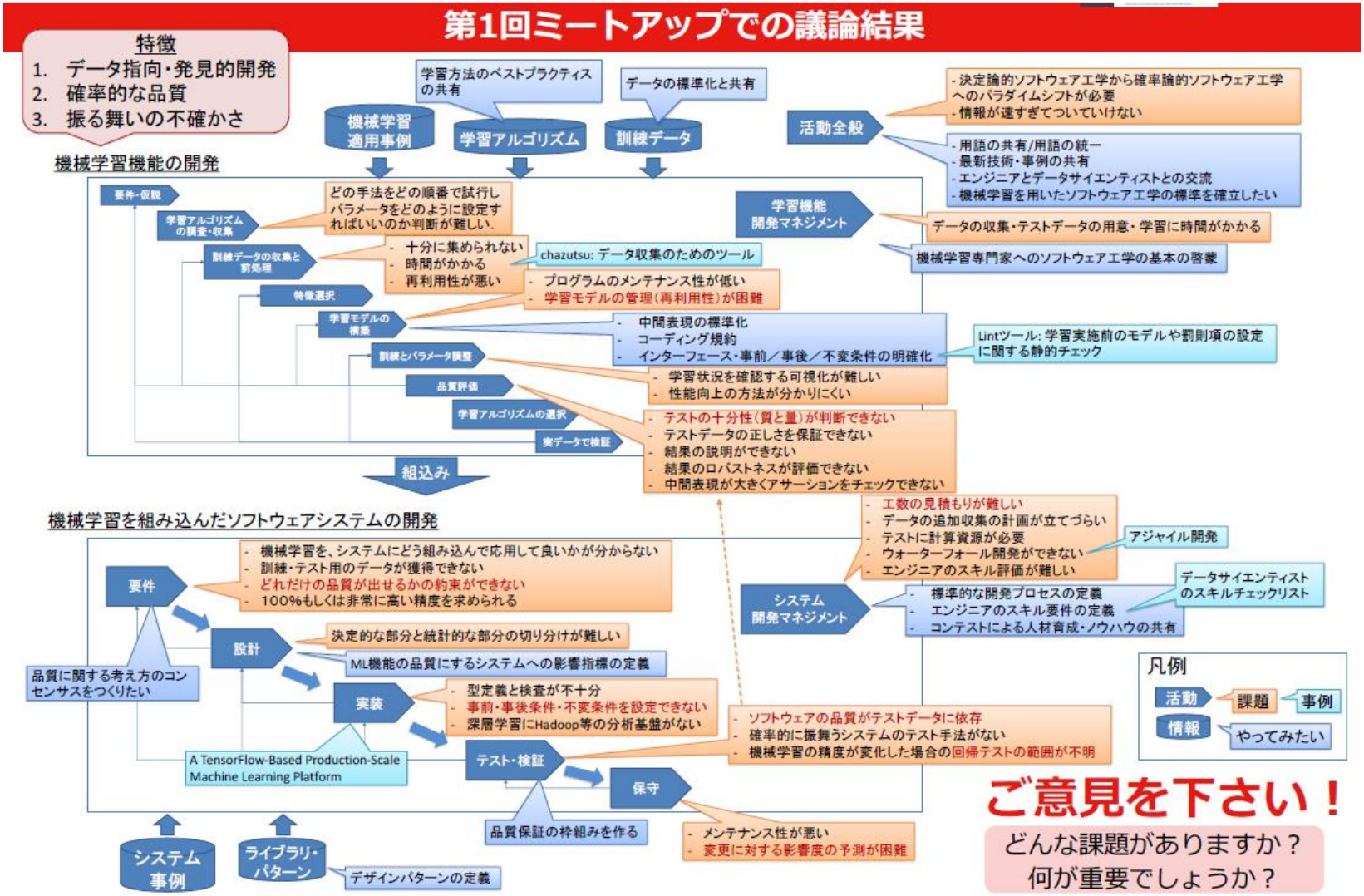
機械学習（帰納的システム開発）

- 大変なこと（「すごいこと」は既知として）
  - 原則として機能は不完全（100%正解は出せない）
  - どの程度の性能が出るか作ってみるまでわからない
  - 大量かつ「適切な」訓練データが必要
  - 訓練データがあればうまくいくとは限らない
  - 訓練データ外のデータ，テストしていないデータでどう振る舞うかは未知（かなり似たデータでさえ）
  - ある出力がなぜ起きたのかは説明できないことが多い（その情報がない，あっても複雑で把握できない，人が言葉にできない傾向をとらえていることも）

「機械学習の技術」→「機械学習工学の技術」

- ソフトウェア工学領域から考えると（ごく一部）
  - 要求工学：ユーザや発注者の「ゴール」を引き出し（ともに創り），今回作るシステムの要求（範囲や制約）を定義，合意，記述し，その妥当性確認をする
  - 設計：保守性など内部品質も含め様々な品質を実現するための適切なアーキテクチャや実現手段を定める
  - テスティング・検証・品質保証：不具合（バグ）に代表される様々な品質の問題を検出・修正し，必要な品質が実現されていることに一定の確信を持てるようにする
  - 運用，保守（適応・進化），管理，プロセス，・・・

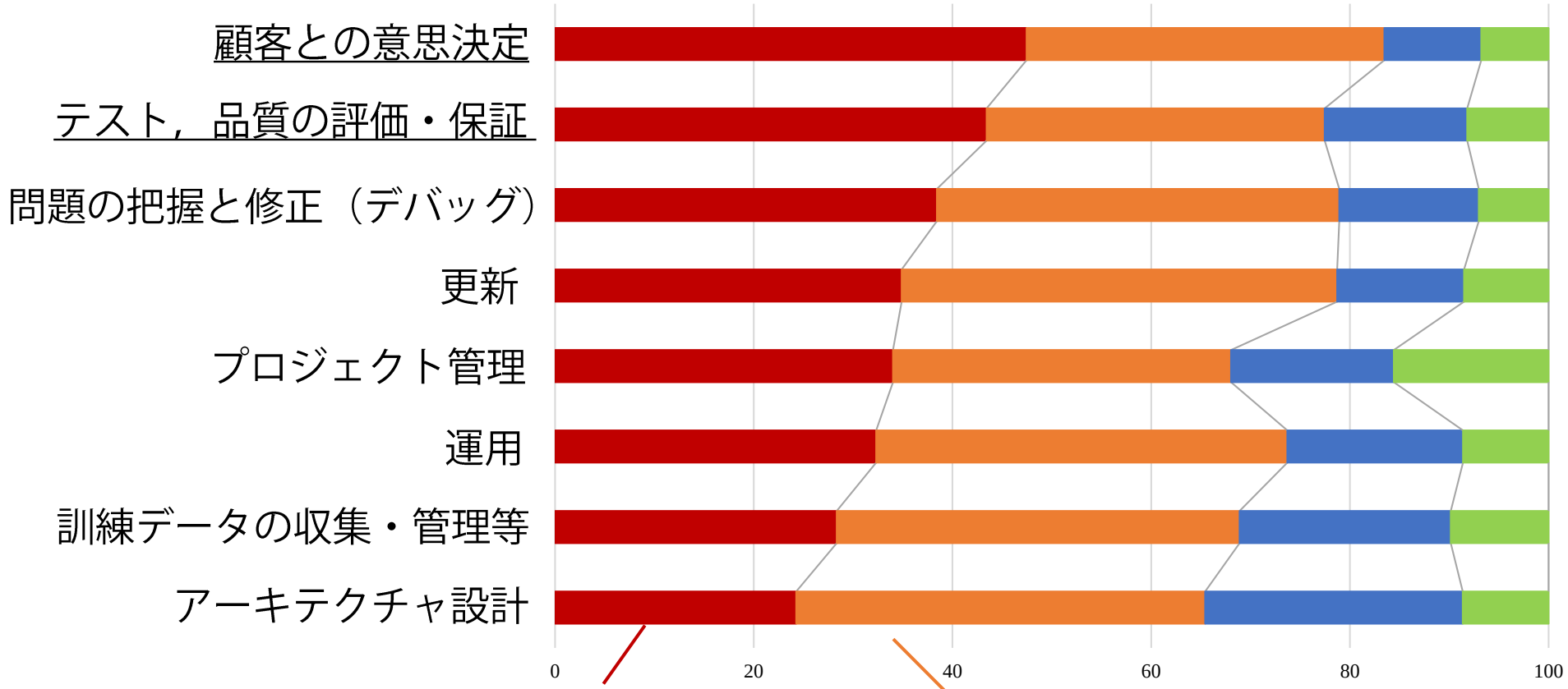
2017年秋のワークショップより（MLSE）



[吉岡ら, SEチャレンジ: 機械学習 x ソフトウェア工学 = 機械学習工学, 2017]
[http://research.nii.ac.jp/~f-ishikawa/work/fose17/]

2018年のアンケートより（MLSE）

- 280名弱のアンケート対象者，大半は開発者
  （ソフトウェア開発経験豊富，機械学習には新規参入）



根本的に異なる新たな考え方が必要

考え方は同じだが
手法／ツールが未成熟

[https://sites.google.com/view/sig-mlse/] → 「発行文献」

課題例： 契約・仕様・受入れの課題

- 「何ができるか」具体的には事前に約束できない
  - 精度がどれだけでるかは基本作らないとわからない
  - 「一緒にがんばりましょう」（準委任）
  - 「後出し」の文句を　しやすい（特に開発者側は大変）
- POC（Proof-of-Concept）
  - まず「お試し」，そこで止まることが多い
  - データの量や前処理を絞ると，さらに結果が出にくい
- 「不確かさ」が多く，顧客・ユーザに「納得」してもらうことが従来よりも困難に
  - 「この予測失敗はどうして？この種のものは直せる？」
  - 「100%当ててくれないと意味がない」（これは論外）

課題例： 保守

- 従来のソフトウェアとはだいぶ性質が異なる「技術的負債」の存在

[Sculley et al., Machine Learning: The High-Interest Credit Card of Technical Debt, 2014]

Googleでの経験から14個の負債を紹介
「機械学習は高利息クレジットカード」

「取り急ぎリリースへ（整理などを後回し）」の弊害や時間経過での劣化が大きく，従来と種類・対応法が異なる

- 例： 一つの入力の傾向が変わると全てが変わる可能性がある（「ここだけ変えれば済む」と言えない）

➡ ***「何とかVer. 1.0リリース」は借金の始まり・・・***

課題例： 自動運転での認識における不確かさ

どれだけの「歩行者」を考える？

「歩行者」の定義は？

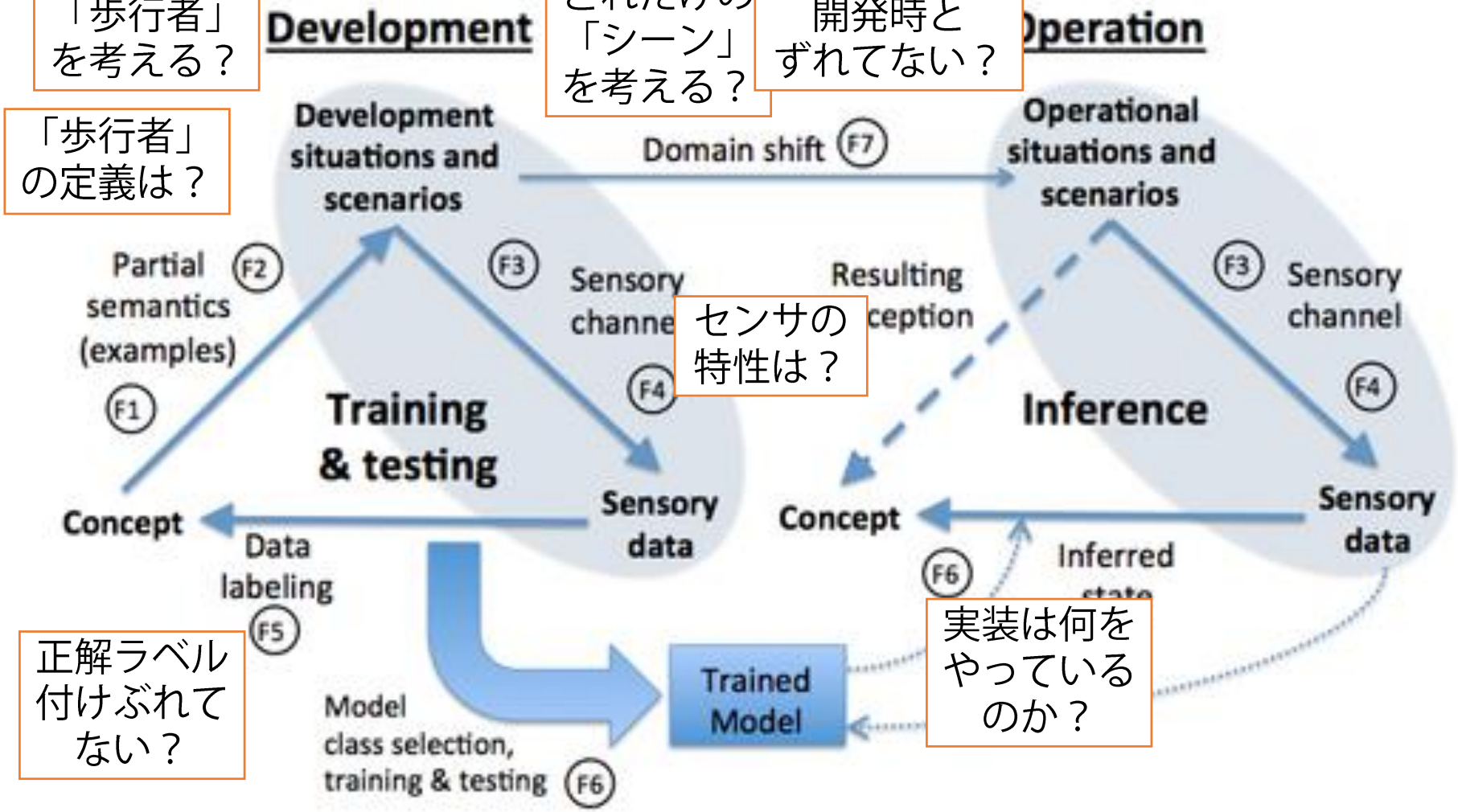
どれだけの「シーン」を考える？

運用時環境は開発時とずれてない？

センサの特性は？

実装は何をやっているのか？

正解ラベル付けぶれてない？



[https://uwaterloo.ca/waterloo-intelligent-systems-engineering-lab/projects/assuredai-safety-assurance-ai-based-automated-driving]

課題例： SQuaREの適合

- ソフトウェアの品質標準をどう拡張すべきか？

  ※ やはりSQuaRE (ISO/IEC 250XXシリーズ)

  - 例：目的，タスク，機能などは，自然には項目化されないので「試験性」などを測る際は，何をどう数えるのか決める（例：「霧の日の画像はテストした？」）
  - 例： 「保守性」をニューラルネットワークに対して事前に（変更を試すことなく）測る術は未確立
  - 例： 高度な AI によっては，「使用性」というよりも，「協調可能性」を評価する必要がある
    （自律的にやってくれるが，把握・制御・信頼できる）

[Kuwajima et. al., Adapting SQuaRE for Quality Assessment of Artificial Intelligence Systems, 2019]

「IoT/AI時代」？

「システムが何をするか」のそもそも

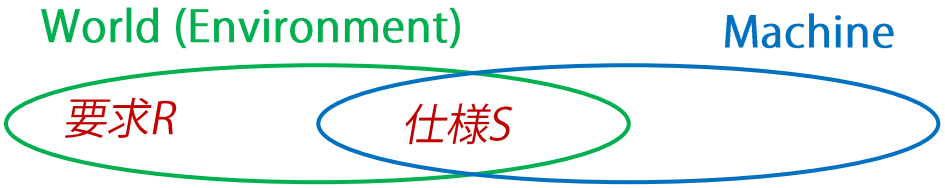
- Zave/Jacksonのモデル
  - 「要求」と「仕様」を別の概念として扱う
  - 「要求」は実現したいことがらであり，
    実現しようと決めたもの
    要求はシステムの制御対象外の領域にも存在する
  - 「仕様」はシステムの振る舞いが満たすべき制約を
    定めたもの（HowではなくWhat）
    仕様はシステムと制御対象世界との境界を定める

    ※「要求仕様」「要件」「ゴール」などの用語の
    使い方は定義・文脈で様々（単に適当なことも）

「システムが何をするか」のそもそも

■Zave/Jacksonのモデル

$$S \quad \models R$$



マシン（開発の成果物・システム）の仕様 S は，

要求 R を満たす？

[Zave et al., Four Dark Corners of Requirements Engineering, 1997]

■例：

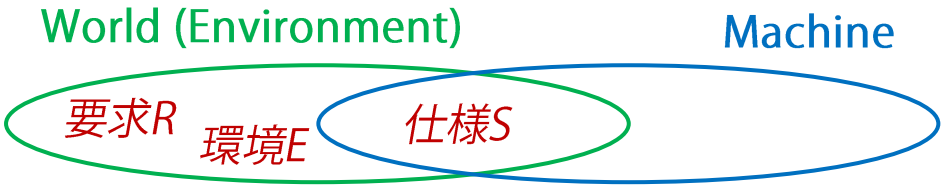
R	• enter 発生回数 ≦ pay 発生回数
S	• pay の発生を検知したら，unlock する • push の発生を検知したら， lock する

[画像： http://www.fujitaka.com/]

「システムが何をするか」のそもそも

■Zave/Jacksonのモデル

$$S,\ E \models R$$



マシン（開発の成果物・システム）の仕様 S は，
想定環境（ドメインの性質） E の下で要求 R を満たす

[Zave et al., Four Dark Corners of Requirements Engineering, 1997]

■例：

[画像： http://www.fujitaka.com/]

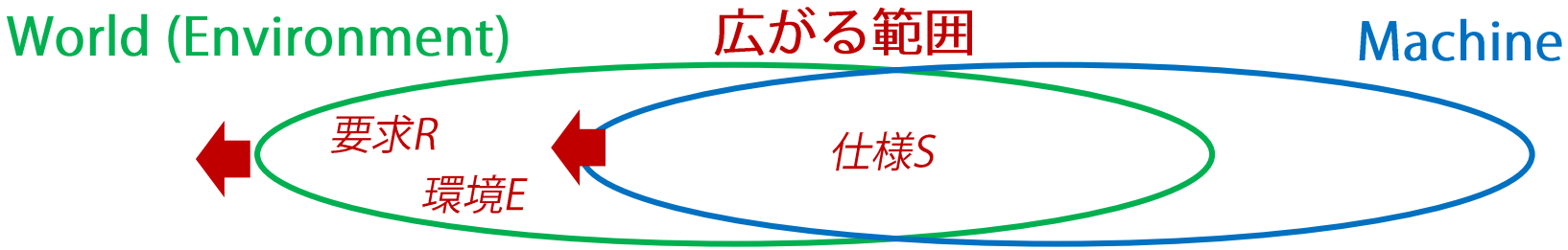
R	• enter 発生回数 ≦ pay 発生回数
S	• pay の発生を検知したら，unlock する • push の発生を検知したら， lock する
E	• push と enter は交互にしか起きないと仮定 • lock が起きてから unlock が起きるまでは push は起こせないと仮定

「IoT・AI時代」の大きなポイント（の一つ）

- 「超スマート社会」「Society 5.0」・・・
  - 物理世界に対する検知・制御能力が増大（IoT/CPS）
  - 演繹的にロジック・ルールが書き出せない
    識別・判断・予測等の機能を実現可能に（AI）

➡ソフトウェアが実世界・人の感覚に
より深く踏み込むように

  - 仕様Sが扱う範囲が広がるということ
  - 要求Rもより広範囲・高度なものを扱うことになる

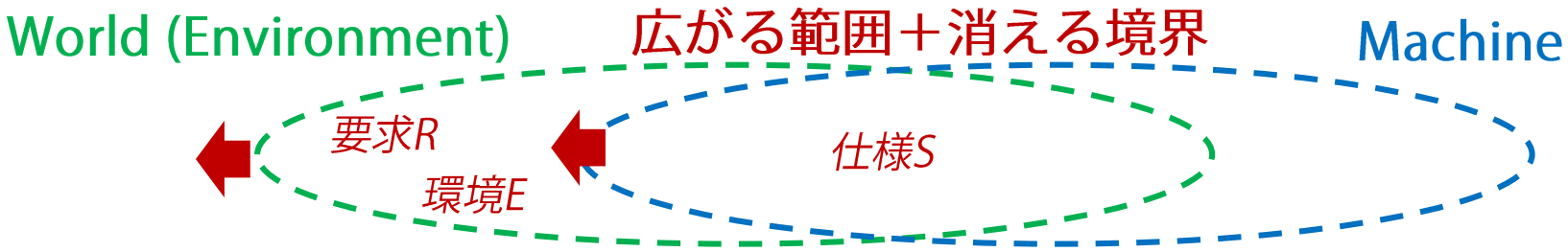


「IoT・AI時代」の大きなポイント（の一つ）

- ソフトウェア制御の範囲（Sの範囲）が広がり，
  実世界における真のRを直接扱うことに
- RとEにおいて「ここまで扱う」という境界が消え
  完全・網羅的な列挙，把握や予測は不可能に
  - 例：「様々な」歩行者を識別する
    （押し車付きでも，集団でも，コスプレでも，・・・）
  - 例：「適切に」運転する
    （安全，快適，マナー遵守など，って具体的には？）
  - 例：「適切に」給与判断する
    （男女や人種の差別はしないとして，何はすべき？）

「IoT・AI時代」の大きなポイント（の一つ）

- オープン性・不確かさとの戦いということ
  - 「完全な機能」「リスクゼロ」は無理
    （建前としてすら，そう言うことは無責任に）
  - 開発前・運用前に想定を尽くしたり，確度の高い分析をしたりすることが困難で，継続的な修正・更新が前提に
    （known unknown と unknown unknown の存在）
- 機械学習を用いた場合，自分たちが構築したものの仕様S（とその実装）もオープン・不確かに



「AIの品質」へのアプローチ例

性能（精度・正解率）

- 訓練・テストのためのデータ：何を基準とする？
  - 「明示的な選別や設計が必要
  - 例：「差別的な給与判断のデータはないか？」
  - 例：「霧の日の運転画像も入っているか？」
- 不完全なものをどう受け入れ役立てる？
  - 精度は高々90%などで、100%にはなり得ない
  - クリック率，工場回転率などの上位目標を評価
  - ビジョンを基に決断できるか？

    例：「どうせ人だって間違えるところ自動化可能」

    例：「人より悪いが人件費よりずっと安い」

➡どちらも発注者・利用者との協働が必要！

先端企業から出ている原則・指針の例

「仮定・想定をテストせよ」は方向性の一つ

（従来の「テスト」より広義）

- ベストプラクティス集
  - 例： データの統計を追跡するなどして，問題として顕在化しない失敗を見張れ

[Zinkevich, Rules of Machine Learning: Best Practices for ML Engineering, 2016]

- どれだけ「テスト」できているかの評価スコア
  - 例： 個々のfeatureの入力値範囲や分布が予想と合うか
  - 例： 直接的なメトリクス（精度等）と実影響のあるメトリクス（クリック率等）との相関はどうか，例えばあえて前者を悪くして比較（A/Bテスト）するとどうなる？

[Breck, What's your ML Test Score? A rubric for ML production systems, 2016]

QA4AIガイドラインの全体像（１）

- 品質を考えるべき対象
  - Data Integrity： データに関する品質
  - Model Robustness： 推論を行うモデルに関する品質
  - System Quality： システム全体の品質
  - Process Agility： プロセスの迅速さ
  - Customer Expectation： 顧客による期待の高さ

  ※ 探索・向上し続けていくプロセス・体制も重要

  ※ 顧客の期待とのバランスが重要
  （期待値コントロールがしっかりなされている）
- 技術カタログ
  - テスト技術など（簡単に後述）

[http://www.qa4ai.jp/]

QA4AIガイドラインの全体像（2）

- 各ドメインでの踏み込んだ分析
  - 自動運転： 不確かさの基でのリスク軽減プロセス
  - 産業プロセス： ステークホルダーの多様性，環境への強い依存性，顧客の納得の必要性
  - スマートスピーカー： 異なる抽象度の要求，自然言語入力の多様性
  - 生成系システム： 自然さなど，AIで実現されるような高度な品質評価機能

[http://www.qa4ai.jp/]

AIに対するテスト技術

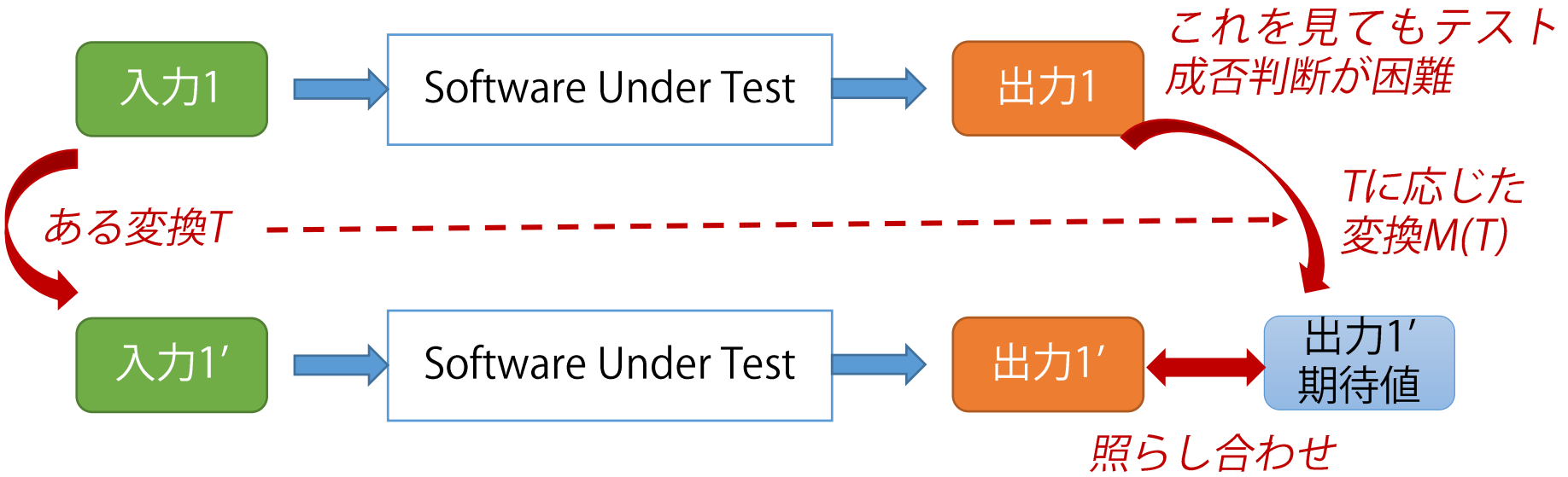
AIに対するテストの課題

- そもそも「テスト不可能」 [Weyuker, On Testing Non-Testable Programs, 1982]
  - 画像分類など，正解を与えることのコストが大きい
  - 給与判断など，唯一の正解を決めがたい場合もある
  - 推薦やデータマイニングなどの場合，未知の正解を求めることが目的で，出力の期待値は存在しない
- 「テストでバグを見つける」？
  - 分類や予測が100%成功することは本来ないため，「Failならバグの存在を示唆」ということにはならない
- 単体テスト？
  - 実装は大きなブラックボックスであり，「一部分だけテストしてデバッグを容易にする」ことができない

背景技術：メタモルフィックテスティング

- メタモルフィックテスティング
  - 多数の入力を用いてテストをしたくても，各出力の正しさ（テスト成否）を判定するのが困難または高コスト

➡ 「入力を変えると出力はこう変わるはず」という関係を検証，既存テストケースから多数のテストケースを生成



例： $\sin(x) = \sin(\pi - x)$

[Segura et al., A Survey on Metamorphic Testing, 2016]

メタモルフィックテスティングの適用

- 訓練アルゴリズム，訓練済みモデル，システム全体のテストへの適用事例
  （ある入力で出力を出してみた後に）
  - ランキング生成： 入力購買データから1位商品を抜く
  - 時系列分析： 入力信号をすべて定数値ずらす
  - ドローン探索プランニング： 地図を回転させる
  - 画像処理： 画像のRGBを入れ替える
  - ・・・

[Murphy et al., Improving the Dependability of Machine Learning Applications, 2008]
[Jarman et al., Metamorphic Testing for Adobe Data Analytics Software, 2017]
[Lindvall et al., Metamorphic Model-based Testing of Autonomous Systems, 2017]
[中島,データセット多様性のソフトウェア・テスティング, 2017]
[Dwarakanath, Identifying Implementation Bugs in Machine Learning, 2018]

背景技術： サーチベースドテスティング

- サーチベースドテスティング
  - 最適化技術（メタヒューリスティック）を用い，
    「欲しいテストケース・テストスイート」を表すスコア
    を最大化するようテストスイートやテストケースを生成
    - 例：「車が人に最も近づくような危ういテスト」ケースを生成
    - 例：「前バージョンと比べ出力が大きく変わる」入力を発見
    - 例：「高カバレッジで小さい」回帰テストスイートを生成

[S. Ali et al., Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation, 2010]
[http://www.evosuite.org/]

これらの適用例

- 機械学習モデルの「あら探し」テスティング
  - 既存画像に雨や画像回転などのノイズを加えた様々な入力を探り，「よい」テストスイートを見つける（サーチベースドテスティング）
  - 目的１：「ニューロンカバレッジ」を最大化することで，多様な振る舞いを促すようにする
  - 目的２：「悪いケース」を検出する
    - 他バージョンとの比較
    - 入力へのノイズ追加でハンドル角が大きく変わるケース（メタモルフィックテスティング）

1.1 original

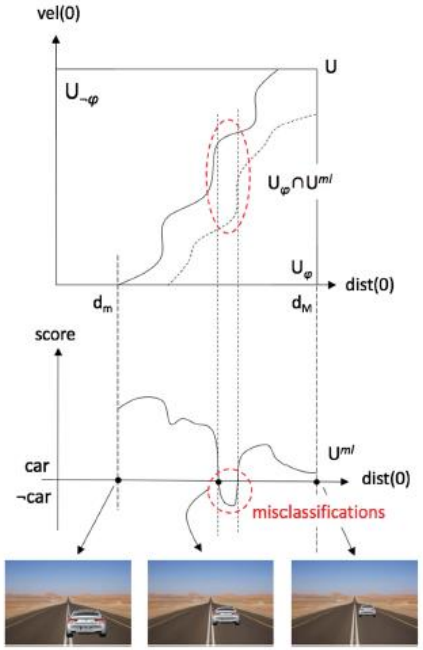
1.2 with added rain

[Pei et al., DeepXplore: Automated Whitebox Testing of Deep Learning Systems, 2017]

[Tian et al., DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars, 2018]

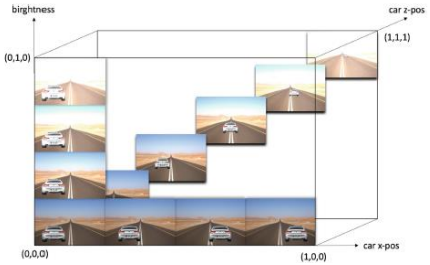
システムレベルの要求に基づくテストの例

- システム全体の要求を踏まえるのが重要？
  - 機械学習部品（生成されたモデル）の精度だけむやみに突き詰めてもしょうがない
  - 例： 遠くの物体を誤認識しても衝突には至らないかも



例題：自動ブレーキシステム（人の操作や先行車の位置が入力，「先行車との距離が一定以上ある」ことが要求）

1．完璧な機械学習部品を用いると要求を満たすが，常に失敗する機械学習部品を使うとそうならないような入力パラメータ領域を絞り込む



2．その領域に限定して機械学習部品が誤認識するような画像を探す

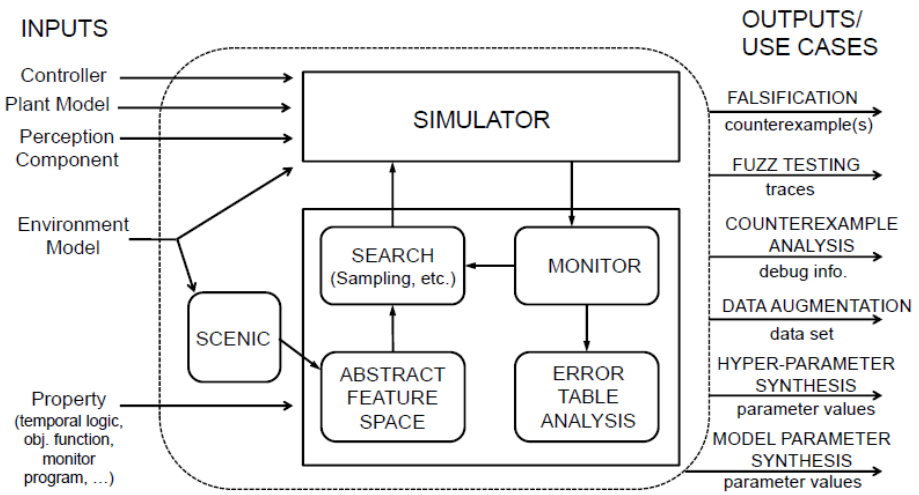
3．その画像を用い要求を満たさないケースを探す

[Dreossi et al., Compositional Falsification of Cyber-Physical Systems with Machine Learning Components, 2017]

補足： システムレベルでのテスト環境

- VerifAI (Berkeley)

システム全体の
物理シミュレーション，
反例探索（後述）などと連動

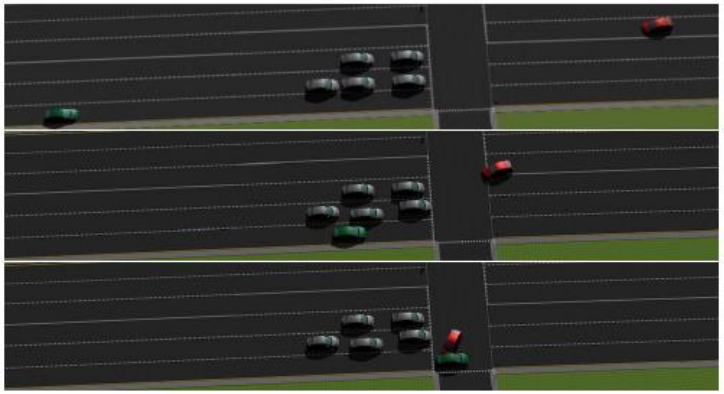



```
# Car going straight
ego = Car on egoLane.median

# Car turning left
Car on leftTurnLane.median

# A car blocking the Ego's view
spot = OrientedPoint on blockLane.median
laneNoise = (-0.5, 0.5)
Car at spot offset by laneNoise @ 0

# Another car 5-8 m behind that
Car at spot2 offset by laneNoise @ (-5, -8)
```

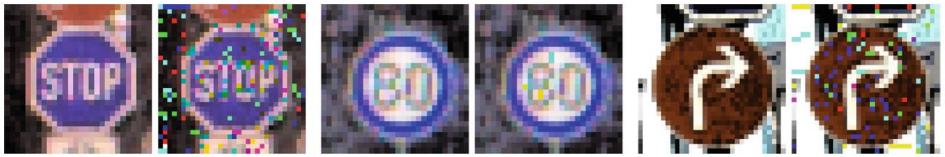


テキスト言語で「シーン」を記述し
CGを合成

[https://github.com/BerkeleyLearnVerify/VerifAI]

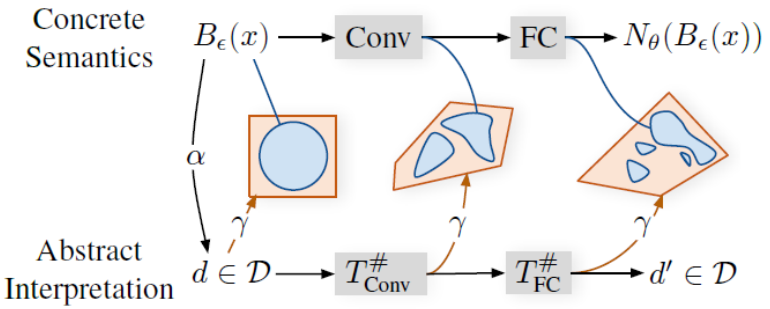
形式検証技術の適用

- 形式検証技術による頑健性検証
  - 画像認識において，「一定範囲の画像操作」を行っても認識結果が変わらないかどうかを網羅的に検証
  - SMTソルバーや抽象解釈を応用



"stop" to "30m speed limit"　"80m speed limit" to "30m speed limit"　"go right" to "go straight"

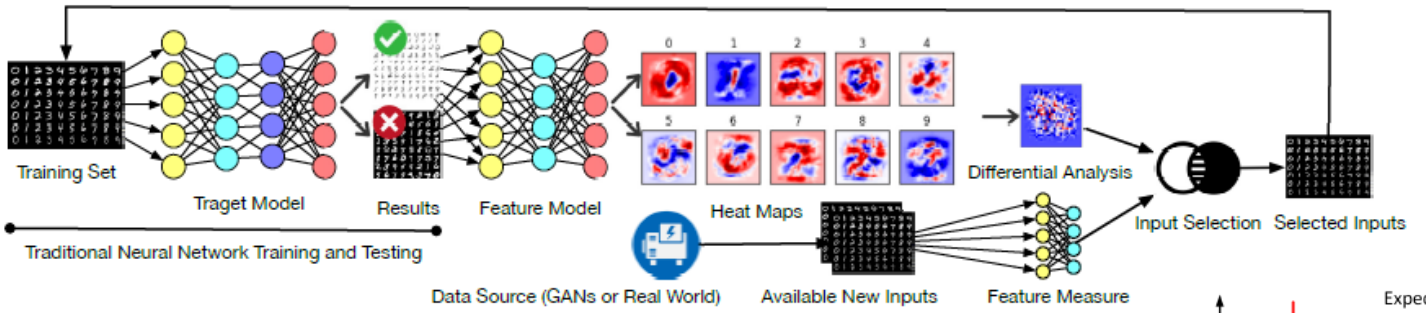
[Huang et al., Safety Verification of Deep Neural Networks, 2017]



[Mirman et al., Differentiable Abstract Interpretation for Provably Robust Neural Networks, 2018]

デバッグの試み

- 識別成功データと失敗データの特徴量を比較し再訓練に用いるデータを選択
  - その他，DNNのどのレイヤで過学習・未学習の影響が現れるかを調べそこを重点的に修正



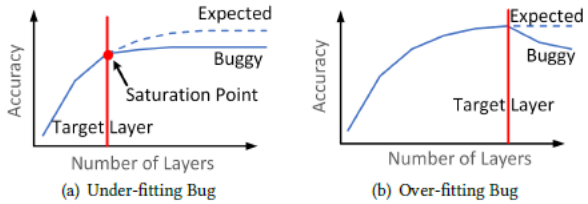


Figure 7: Test Accuracy v.s. Number of Layers

[Ma et al., MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection, 2018]

AIによるテスト？

サーチベースドテスティングをそう受け止めてもよい

サーチベースドテスティングのレベル感

- この10年ほど盛ん
  - テスト生成に限らず，サーチベースドソフトウェア工学
- 2018年のJava Unit Testing Competition
  - 比較用に複数ツールを組み合わせた「最強ツール」は，人が作ったものよりよいテストスイートを10秒で生成
  - ここでの評価基準は，コードカバレッジとミューテーションスコア（人工バグの検出率），テストケース数
  - 理解容易性や自然さも考慮することがある

[Molina et al, Java Unit Testing Tool Competition - Sixth Round]
[https://code.fb.com/developer-tools/sapienz-intelligent-automated-software-testing-at-scale/]

Facebookでの事例（１）

- Sapienz：モバイルアプリテスティング

A
テスト数：25
テスト長合計：150
発見クラッシュ数： 5

B
テスト数：80
テスト長合計：500
発見クラッシュ数： 63

C
テスト数：60
テスト長合計：100
発見クラッシュ数： 20

何百・何千サイクル

AとBに含まれるテストケースを
混ぜ合わせた「子ども」を
作ってみよう（遺伝子の勾配）

Cに含まれるテストケースでの入力値を
少し変えてさせてみよう（遺伝子の突然変異）

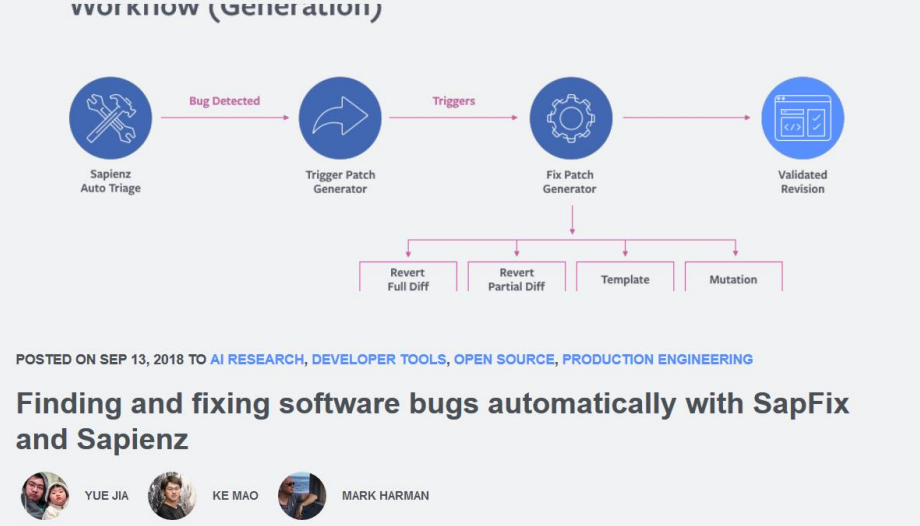
[Mao et al., Sapienz: multi-objective automated testing for Android applications, 2016]
[Alshahwan., et al., Deploying Search Based Software Engineering with Sapienz at Facebook, 2018]

Facebookでの事例（２）

- Facebookが使っている（Continuous Integrationのサイクルに入っている）との発表あり（Sapienz）
  - カバレッジ，テスト長，検出バグ数が指標

[Mao et al., Sapienz: multi-objective automated testing for Android applications, 2016]
[Alshahwan., et al., Deploying Search Based Software Engineering with Sapienz at Facebook, 2018]

- さらに「自動バグ修正」ツールも連動（SapFix）



典型的なバグ修正を
いろいろやってみて，
テストをパスするか見る

[https://code.fb.com/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/]

我々の産学連携研究の一例（１）

- 不安定なパラメータ領域検出（Simulinkモデル）

  パラメータ値の少しの変更が安全や品質に

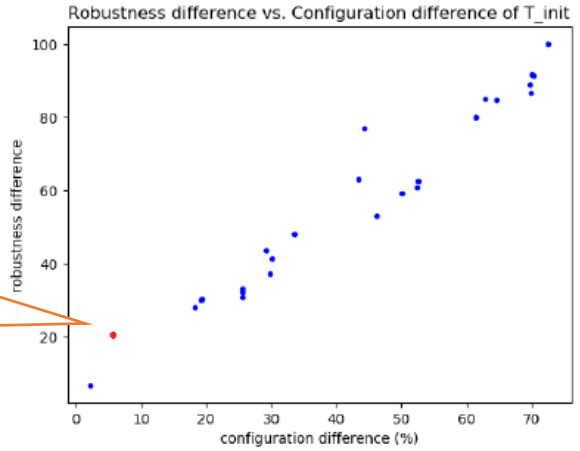
  *大きく影響する*ような状況を報告

  → 「サーチベースド敏感性分析」

  - 設計パラメータ（避けるか重点的にテスト）も，

    環境パラメータ（重点的にテスト）も

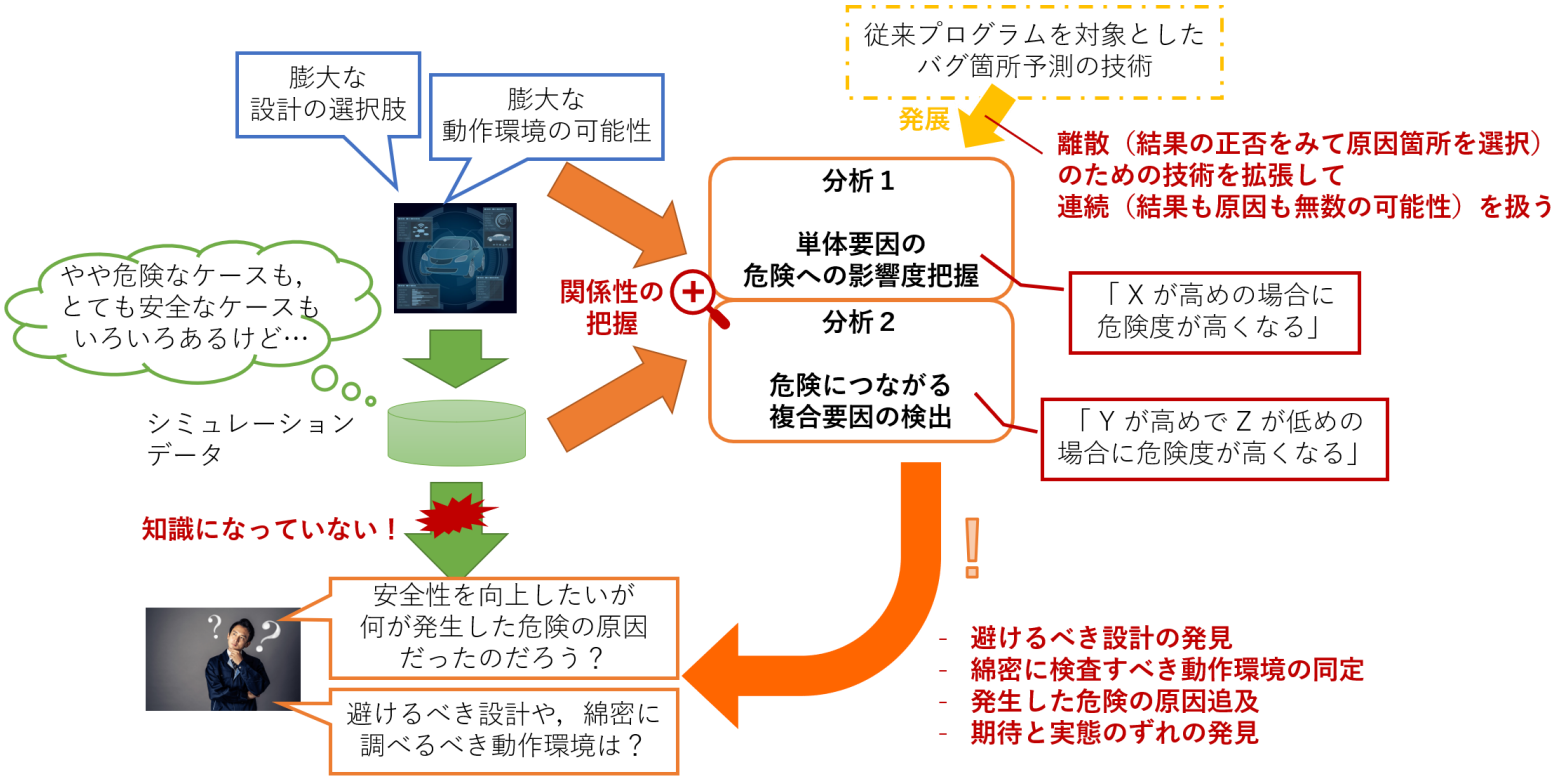
動き出し時間設定の
0.12 秒の差により，
衝突無しの状況が
20km/h衝突に変わる



[Lee et al., Stability Analysis for Safety of Automotive Multi-Product Lines: A Search-Based Approach, GECCO'19]

我々の産学連携研究の一例（２）

- テストログから知識をマイニング
  - もともと「0 か 1 か」の「バグ箇所」を当てる技術
    →より危険性が高い状況に影響している要因を当てる技術



[Zhang et al., Assessing the Relation Between Hazards and Variability in Automotive Systems, ICECCS'19]

我々の産学連携研究の一例（３）

- 自車の経路を最適化により決定する経路計画プログラムのテスト
  - 「危険度が高い」ことを目指すだけでは非現実的なテストシナリオを生成してしまう
    - 極端な例：すべての他車が自社に突撃してくる
  - でも現実的って何？「こういうシナリオは作らない」というテスト要求をすべて書き出せますか？

※ 近日プレスリリース予定

[Caro et al., Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems, ICST'20]

補足：「AI」はもっといろいろある

- 例：リポジトリマイニング
  （MSR: Software Repository Mining）
  - GitHubや組織内リポジトリに対するデータマイニング
  - 例：「このモジュールに高確率でバグあり」
  - 例：「あなたがいじったコードをいじった人は，あのコードも一緒にいじっていますが大丈夫？」
  - 例：「アプリストアで機能追加要望コメント抽出」

[Google bugspots, http://google-engtools.blogspot.com/2011/12/bug-prediction-at-google.html]
門田先生資料（2016）[https://dl.dropboxusercontent.com/u/1350791/jssst2016-msr-tutorial.pdf]

おわりに

「AI時代の品質」改めて（１）

- ビジネス・人の営み・社会における価値を考慮
  - ファジーな本来やりたいこと，仕事そのもののあり方について，ソフトウェアによる実現が可能に
  - 顧客・利用者・開発者（品質保証者）の協働・価値共創
- 速い変化・高い不確実性が原則に
  - 顧客・利用者が望むことを書いてみても，部分的であり現時点での仮説に過ぎない
  - 実装・技術側も含めて「初めて」「作ってみないとわからない」ことに取り組んでいく
  - 要求・環境の傾向変化により大きな影響を受ける（何もしなくても壊れる）

「AI時代の品質」改めて（２）

- プロダクトとプロセス双方の指標定義，継続的な監視・測定，フィードバックが必要
  - 不確かさと変化への対応
- 探索的なものも含めつつ自動化されたテスト
  （実行の自動化は当然として設計・生成を自動化）
  - 煩雑さ・膨大さへの対処
  - 複雑で因果関係がつかめない挙動への対処
- これまでとは異なる設計や問題点に関する知見・経験の形式知化，共有，活用
  - プロジェクト，企業を超えて
  - 記述や構築時の適切な設計，その後の適切な検査・指摘

あれ？何も新しくない？

- ソフトウェアに携わる人々が本来やること
  - しかし今でもやりきれていない
  - AIをはじめこれからのシステムでは逃げられない
  - そちらに集中するためにも，従来から典型的な問題でつまずき続けているべきではない
- 一個人・一企業の経験・知見だけで取り組める時代ではもうない
  （この場の皆さんに伝えることではないですが・・・）

おわりに：AI時代のテスト

目指すべき大原則は昔から共通かもしれないが，
向き合う必要性が増大，技術的実現は大きく変化

- 実世界での意義や安全性，ユーザの感性的な満足の追求
- 探索的・投機的な仮説の設定・監視・評価に基づく意思決定と継続的な改善
- 数学・高度な自動化の理解・活用
- 利用者・発注者・開発者（テスター含む）の協働による価値創造
- ・・・
- といったことへの本気のパラダイムシフト

楽しんで切り拓いていきましょう！