「ソフトウェア工学」

(7) 発展的トピック

東大理学部情報科学科講義

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

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

目次

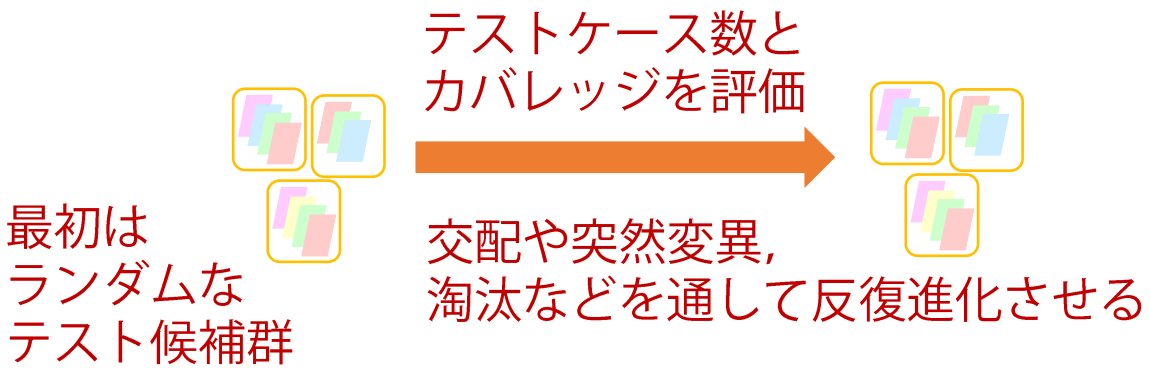
- 様々な研究・発展の方向性
- 機械学習工学

Search-Based Software Engineering

- Search-Based Software Engineering：
  ソフトウェア工学における様々な問題を最適化問題に帰着
  - 「与えられた範囲でいろいろな解を検討し，
    目的となるスコアが最小・最大になるような解を得る」
  - 自動テスト生成，テスト最小化，テスト優先付け，
    自動設計，取り組む要求の選択，不具合修正など
  - 進化計算などメタヒューリスティック最適化技術を用いる
    - 数学的な解析ではなく，「ある解で試してみてスコアを得る」ことを
      反復し，よりよいスコアを得ることを目指す

例：Search-Based Testing

- 例：テスト入力生成の場合
  - メソッド m1(int x, int y) の単体テストのために，テストスイートにおける入力リスト (x0, y0), (x1, y2), …, (xn, yn) を生成したい
  - ブランチカバレッジを最大にしつつ，テストケース数nを最小にする探索・最適化の問題と見なして解く



[https://www.evosuite.org/]
[Fraser et al., Whole Test Suite Generation, TSE'13]

最近の事例：Facebookにおけるモバイルアプリのテスト

- Sapienz：クラッシュを見つけるためのテスト自動生成

テストスイートA
テストケース数：25
コマンド数合計：150
発見クラッシュ数：5

テストスイートB
テストケース数：80
コマンド数合計：500
発見クラッシュ数：63

テストスイートC
テスト数：90
コマンド数：520
発見クラッシュ数：20

＜進化ステップの例＞
淘汰：CはAとBと比べいいところがないので無視

交配：AとBが持っているテストケースを
半々で混ぜ合わせてみよう

最終的には
「コンパクトな
テスト実行レポート」
が得られる

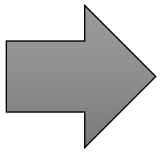
[Mao et al., Sapienz：multi-objective automated testing for Android applications, ISSTA'16]
[Alshahwan., et al., Deploying Search Based Software Engineering with Sapienz at Facebook, SSBSE'18]

デバッグ支援のための自動化技術 (1) 欠陥局所化

- 欠陥局所化（Fault Localization）：バグがどこにあるか推定
  - アプローチの例：Spectrum-based Fault Localization
    - 成功したテストと失敗したテストが実行した文を比べる

テスト結果

l.1	l.2	l.3	…	Result
✓	✓		…	PASS
✓		✓	…	FAIL
		✓	…	FAIL
…	…	…	…	…



「怪しさ」スコア・ランキング

Line	Score
l.3	0.8
l.5	0.72
l.9	0.6
…	…

例：Ochiai指標 $\dfrac{\text{その文を通る失敗したテストケースの数}}{\sqrt{\text{失敗したテストケースの数} \times \text{その文を通る失敗したテストケースの数}}}$

[Wong et al., A Survey on Software Fault Localization, TSE'16]

デバッグ支援のための自動化技術 (2) プログラム自動修正

- プログラム修正（Automated Program Repair）：
  テストをパスするようなプログラムの修正を探索
  - アプローチの例：典型的な修正操作をいろいろ適用してみる
    - > と >= の置き換えなどのミューテーション操作
    - GitHubでマイニングした典型的な修正操作
  - アプローチは多様
  - テストケースが不十分だと「不適切な修正」をしてしまう
    - 極端な例：「クラッシュが起きないこと」しかチェックされないなら，
      1行目に return 文を足して何もしないプログラムにしてしまえばよい

[Gazzola et al., Automatic Software Repair: A Survey, TSE'19]

実証的ソフトウェア工学

- 実証的ソフトウェア工学
  （Empirical Software Engineering）：
  開発現場のデータなど実践から知見を確立し活用していく
  - 「私が考えた最強の開発手法」への反省
  - GitHubなどによる大量データの分析も受け発展
  - 実践データを扱うためデータサイエンス・機械学習の技術が活躍

リポジトリマイニング

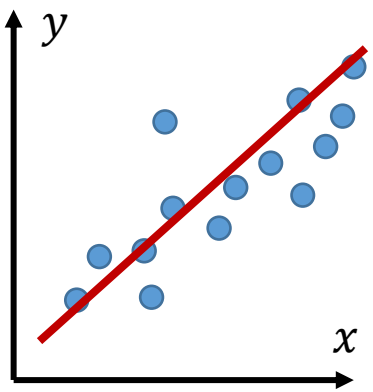
- 例：リポジトリマイニング
  （Mining Software Repositories: MSR）
  - データマイニング技術の活用
  - 例：GitHubのコード履歴やエンジニアの相互作用から，典型的なバグ修正パターンや，却下されがちなPullRequestの特徴を抽出
  - 例：StackOverflow（エンジニアQ&Aサイト）から，誤り・質問が多いAPIやサンプルコードの傾向を抽出
  - 例：アプリストアのコメントから，機能要望や苦情などを分類，アプリごとの傾向を抽出

目次

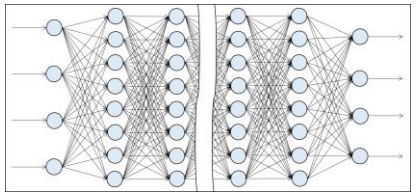
- 様々な研究の方向性
- 機械学習工学

機械学習： とても簡単に

- ある会社での年齢 x のときの給与 y を予測したい
  - $y = ax + b$ と表現できるとして，
    過去のデータと「一番合う」ように
    a, b を決めれば判定・予測プログラムが作れる！



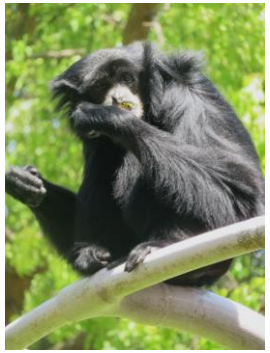
  - 実際は1次関数（パラメータ2つ）では表現が困難
    - ディープラーニング（深層学習）では，より多様な入力形式を扱え，
      何百万以上のパラメータで予測関数を表現する



現在話題になっているAI（人工知能）は主に，
判定や予測の機能をデータからつくる機械学習型

機械学習：別の例

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

画像 =
各点の色を表す
数値の集まり

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

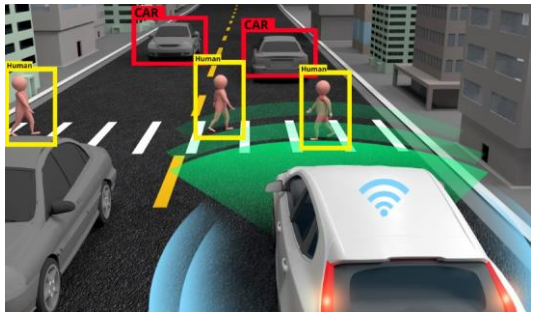
77 81 123	122 158 6	76 63 42
3 3 78	19 183 84	76 63 123
98 83 111	123 7 99	253 48 91

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

この線引きを訓練データから作る

機械学習：すごいこと

- 規則性を書き出すことが困難な機能であっても実現可能
  - 画像における物体の識別
    - 自動運転における歩行者や信号
    - 工場における不良品や機器劣化
    - 医療画像における腫瘍
  - 工場機器の制御
  - ローンや保険に関する判断
  - 音声や文書の分類，検索，翻訳
  - 画像や動画の色塗りや補完，生成



機械学習：大変なこと

- 判別や予測の性能の不確かさ
  - 原則として機能は不完全（100%正解することはない）
  - どの程度の性能が出るか作ってみるまでわからない
- 振る舞いの不確かさ
  - 新たな入力でどう振る舞うかは未知（かなり似たデータでも）
  - ある出力がなぜ起きたのかは説明できないことが多い
- データへの依存性
  - 大量かつ「適切な」訓練データが必要
  - 基本的にデータに対して相対的な評価しかできない

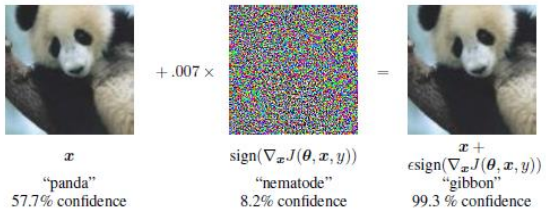
課題の例：技術的限界・性能や振る舞いの不確かさ

- Googleフォトの画像認識
  - 黒人の写真を「ゴリラ」とタグ付け
  - 2年経って本質的には直せず（ゴリラを禁止ワード扱いに）

[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]

- 優れた画像識別器が微小ノイズで誤認識
  （敵対的サンプル）



「パンダ」が「テナガザル」に

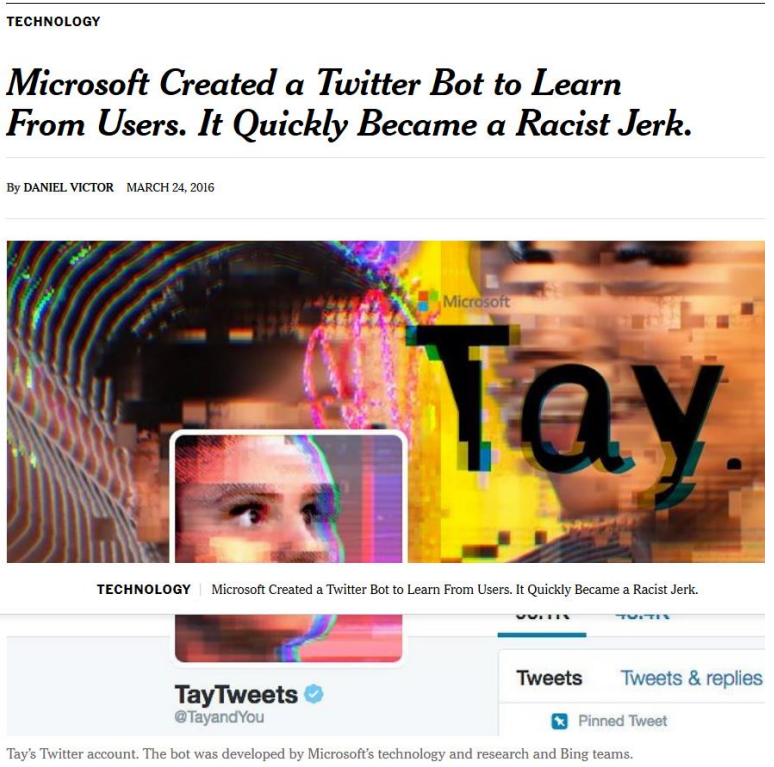
物理的なテープ貼付による誤認識

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

[Ackerman, Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms, IEEE Spectrum'17]

課題の例：訓練データ・攻撃・継続的学習

- 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
  - 継続的に学習・更新し続けるものの監視や制御の問題
    （もしも攻撃がなかったとしても）
  - 人間・社会の要請に事前に・事後に
    どう応えるか


TECHNOLOGY

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

By DANIEL VICTOR MARCH 24, 2016

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

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

[https://www.nytimes.com/2016/03/25/technology/microsoft-created-a-twitter-bot-to-learn-from-users-it-quickly-became-a-racist-jerk.html]
(access: 2021/09/27)

課題の例：バイアス・倫理的要求

- 社会的に望ましくない差別的なバイアス（偏り）の発生
  - 人間の（ときには無意識な）差別を学習
  - 少数例外に対しては学習しきらず予測性能が低くなりがち

女性に不利な雇用判断
→
過去の差別を学習？女性が少数データ？

[https://jp.reuters.com/article/amazon-jobs-ai-analysis-idJPKCN1ML0DN]
(access: 2021/09/27)

アフリカ系の名前で検索すると
逮捕歴データベースの広告
→
ユーザの潜在意識によるクリックを反映？

[L. Sweeney, Discrimination in Online Ad Delivery, ACM Queue'13]

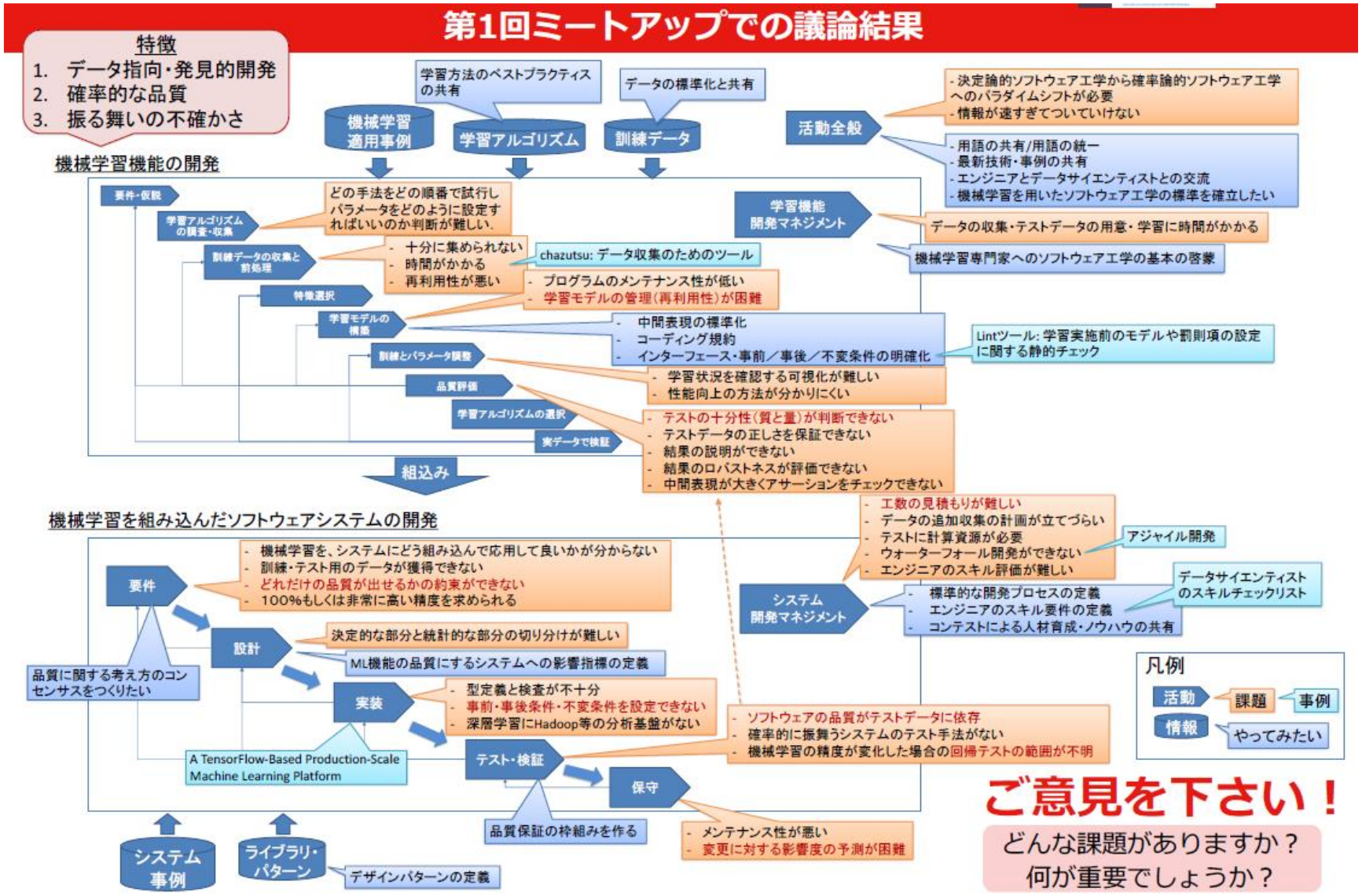
課題の例：特定の状況・システム全体のデザイン

- Tesla車における一連の事故（現在調査中）
  - 緊急車両がかかわるシーンでの12件の衝突
- 米国運輸省道路交通安全局からの提出要請
  - 発煙筒やコーン，路肩位置の人や車両など
    緊急事態を識別するための仕組み
  - 特に暗い状況の考慮
  - シミュレーションデータや訓練データも含めて
    リリース前に採った施策・検証・確認の内容

[https://www.nhtsa.gov/recalls?nhtsaId=PE21020]

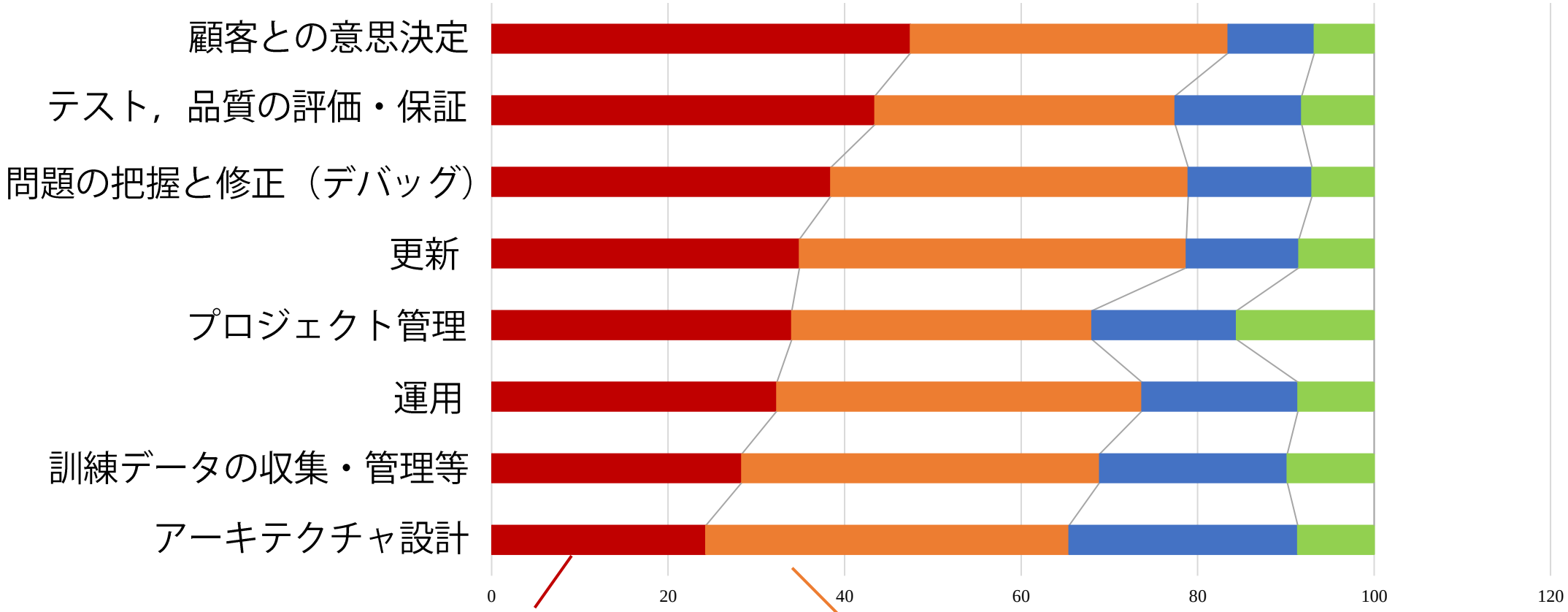
2017年秋のワークショップより

[http://research.nii.ac.jp/~f-ishikawa/work/fose17/]



2018年のアンケートより（MLSE研究会）

■280名弱のアンケート対象者，大半はソフトウェア技術者



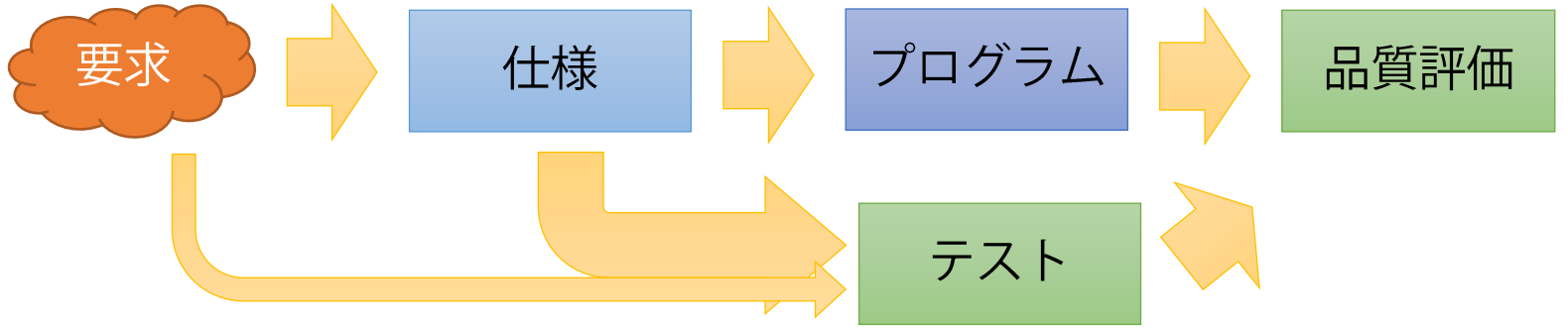
根本的に異なる新たな考え方が必要

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」

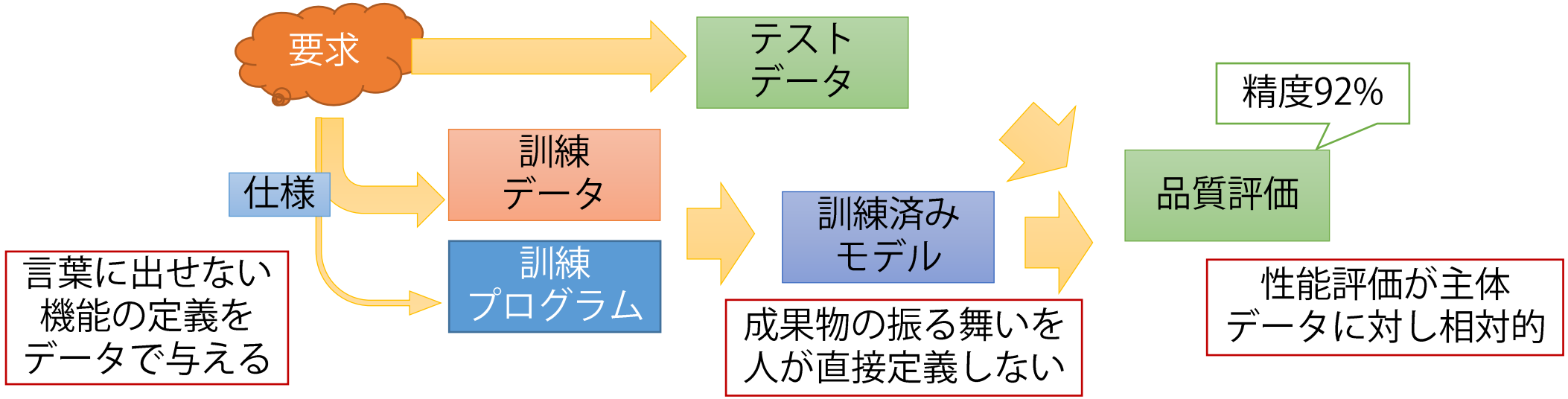
従来ソフトウェアと機械学習：構築手段の視点

従来ソフトウェア



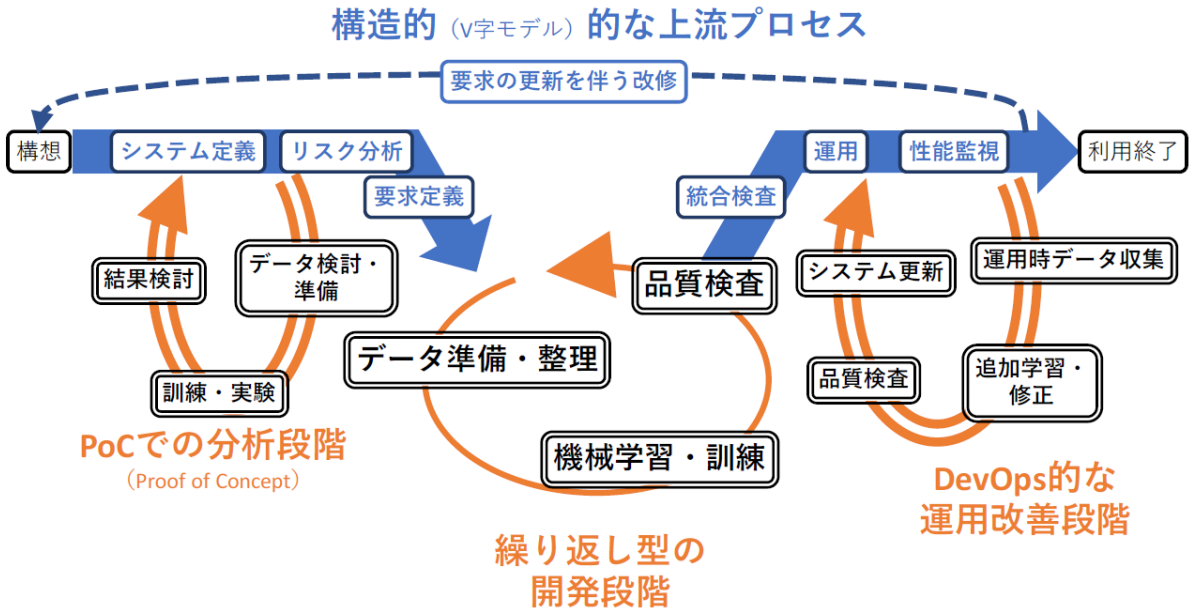
※ 工程はかなり簡素化
※ 実際のシステムでは両方を統合

機械学習型AI



従来ソフトウェアと機械学習：プロセス全体の視点

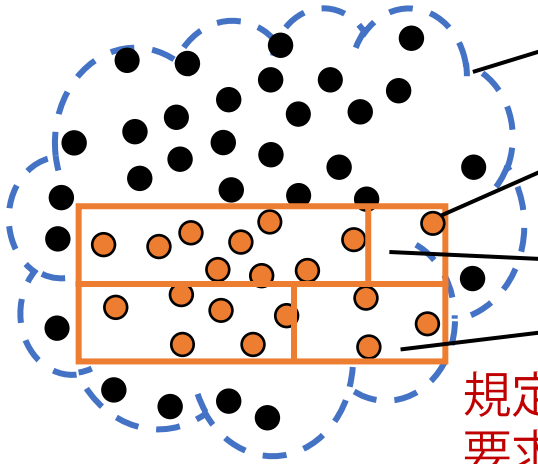
- V字モデルと比べて
  - 事前に「実現可能性などの検証」が必須
    （PoC: Proof of Concept）
  - 実装工程は，試行錯誤の反復
  - 運用時に，データ傾向変化
    などによる性能劣化の監視や，
    新しいデータでの再学習が必要



[機械学習品質マネジメントガイドライン 2.1.0] より

従来ソフトウェアと機械学習：要求充足の視点

従来ソフトウェア



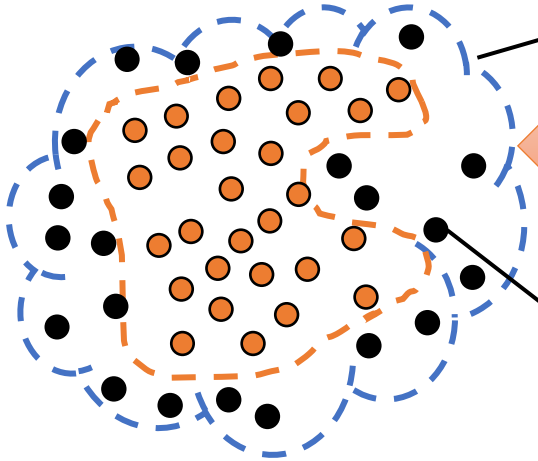
ゴール：名医のように健康状況を適切に判定する

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 6.5 以上の場合…
- 面積を基にして…アルゴリズムで影の有無を判定し…

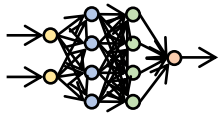
規定された
要求範囲だけ確実に充足

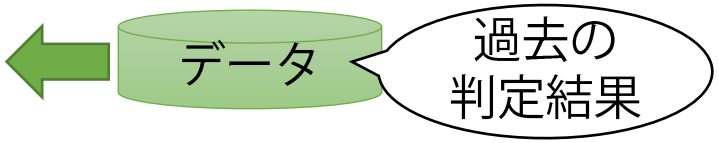
機械学習型AI



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%





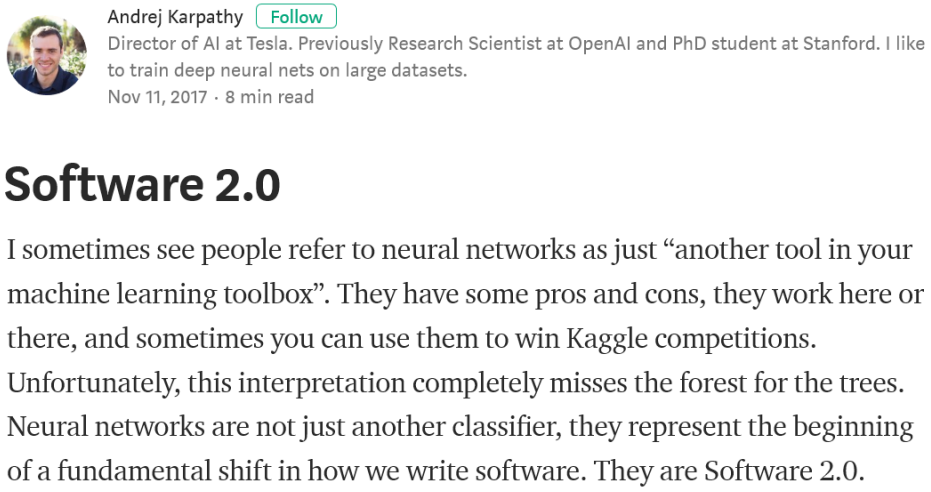
ゴール全体を
大まかに達成

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するかも）

従来ソフトウェアと機械学習：実装の視点

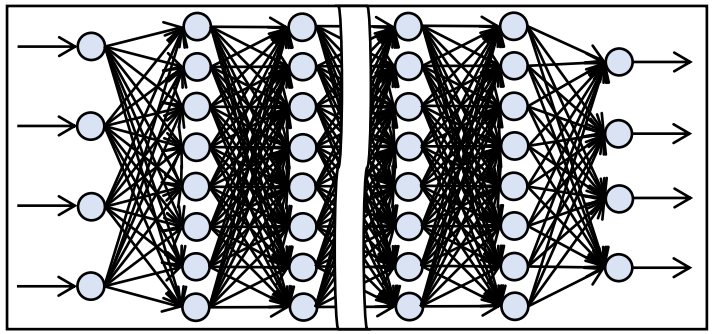
- 成果物となるソフトウェアは，
  大量のパラメーター値を自動探索・設定して作成
  - 特に深層学習（Deep Neural Network）の場合，
    何百万・何千万といったパラメーター数



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]
(access: 2022/07/20)

課題例：契約・仕様・受入れ

- 「何ができるか」，特に予測性能は事前に約束できない
  - 「一緒にがんばりましょう」（準委任・発注側が大変）
  - 「後出し」の文句をしやすい（開発側が大変）
- POC（Proof-of-Concept）貧乏
  - まず「お試し」，そこで止まることが多い
  - データ収集や前処理の工数を絞ると，さらに結果が出にくい
- 「不確かさ」が多く，顧客・ユーザの「納得」がより困難
  - 「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」（無理解）

課題例：保守

- 従来のソフトウェアとは性質が異なる「技術的負債」
  - *※「整理などの後回し」の弊害・時間経過での劣化*
  - 入力分布やハイパーパラメーターが絡み合って個々が全体に大きな影響（Changing Anything Changes Everything）
  - 暗黙的に変化しやすいデータへの強い依存性：学習時と運用時の分布変化も（Concept drift / covariance shift）
  - 利用パッケージ多数を橋渡しする「パイプラインジャングル」コード，さらに大量の試行錯誤・実験の一時コード

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」

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

重要な用語：ドリフト

- データドリフト
  - 入力データの統計的な分布が訓練時と運用時で変わる
  - 例：工場での不良品検知において，訓練時には少量だったタイプの不良品が多数発生し，訓練時にはなかった暗い入力画像が多い
- コンセプトドリフト
  - 入力と出力の関係性が訓練時と運用時で変わる
  - 例：今までは「不適切なコンテンツ」と分類されていなかった投稿が，不適切と分類すべきだと変わった

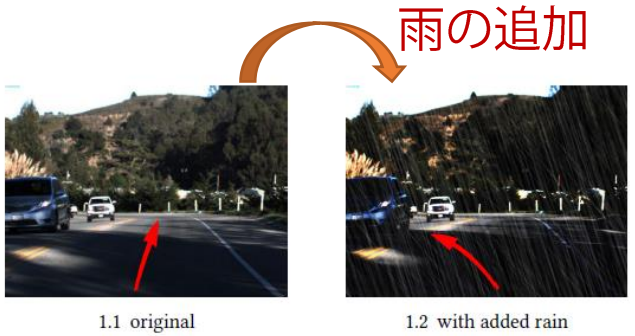
問題例：テスト

- テストオラクル（期待値の設定）？
  - 画像分類など，正解を与えることのコストが非常に大きい
  - 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は，本来正解はない
- 「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，
    「結果が合わなければバグが存在」とはならない
- 単体テスト？
  - 一つの実装が様々な条件下で様々な対象を扱う（分割統治困難）

技術の一例：初期のテスティングツール研究

- 深層ニューラルネットワークに対する「ホワイトボックステスティング」ツール
  - 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，テストスイートを最適化により自動生成
  - 目的１：「ニューロンカバレッジ」を最大化，多様な挙動を促す
  - 目的２：「望ましくない挙動」を最大化，問題に至る入力を検出
    - 例：「入力にノイズが加わったときに，出力のハンドル角があまりにも大きく変わる場合」を検出（出力の変化量を最大化するノイズを探る）

      ※ メタモルフィックテスティングと呼ぶ擬似的なテスト判定基準



[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]

補足：機械学習工学

- 以下の資料などでもう少し動向など紹介
  - テスト・検証周り
    https://www.slideshare.net/FuyukiIshikawa/ss-238917972
  - 機械学習工学全般
    https://www.slideshare.net/FuyukiIshikawa/ses20-238448039
  - 書籍

まとめ

- ソフトウェア工学は今でも進化中
  - 計算の高速化，基盤技術の進化も受けた多様な自動化技術の追求
  - 研究の信頼性・再現性がより強く求められるとともに，
    大量データが得られる中でデータ駆動の知見の追求
  - 現在は，機械学習・AIのための技術適合・深化の追及が特に盛ん