テスト技術者資格制度

Foundation Level シラバス

AI テスティング

Verison1.0.J01

International Software Testing Qualifications Board

提供

Alliance for Qualification、Artificial Intelligence United、
Chinese Software Testing Qualifications Board。
および韓国ソフトウェアテスト資格認定委員会




著作権について

改訂履歴

バージョン	日付	備考
1.0	2021/10/01	GA 用リリース

JSTQB

バージョン	日付	備考
1.0.J01	2023/02/25	日本語版リリース

目次

謝辞

この文書は、2021 年 10 月 1 日 st®に ISTQB の総会で正式に発表された。

これは、以下メンバーで構成される International Software Testing Qualifications Board のチームが作成した：Klaudia Dussa-Zieger（議長）、Werner Henschelchen、Vipul Kocher、Qin Liu、Stuart Reid、Kyle Siemens、Adam Leon Smith

チームは、シラバスの作成に貢献した以下 3 つの著者（団体）に感謝している：

- A4Q：Rex Black, Bruno Legeard, Jeremias Rößler, Adam Leon Smith, Stephan Goericke,Werner Henschelchen
- AiU:主著者：Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni and Sonika Bengani
  共著者：Rik Marselis, José M. Diaz Delgado
- CSTQB/KSTQB: Qin Liu, Stuart Reid

シラバスの作成に協力した Exam Working Group、Glossary Working Group、Marketing Working Group、技術的な編集をした Graham Bath、提案や意見を出した Member Board に感謝する。

このシラバスのレビューとコメントには、以下の方々が参加した。
Laura Albert, Reto Armuzzi, Árpád Beszédes, Armin Born, Géza Bujdosó, Renzo Cerquozzi, Sudeep Chatterjee, Seunghee Choi, Young-jae Choi, Piet de Roo, Myriam Christener, Jean-Baptiste Crouigneau, Guofu Ding, Erwin Engelsma, Hongfei Fan, Péter Földházi Jr., Tamás Gergely, Ferdinand Gramsamer, Attila Gyúri, Matthias Hamburg, Tobias Horn, Jarosław Hryszko, Beata Karpinska, Joan Killeen, Rik Kochuyt, Thomas Letzkus, Chunhui Li, Haiying Liu, Gary Mogyorodi, Rik Marselis, Imre Mészáros, Tetsu Nagata, Ingvar Nordström, Gábor Péterffy,Tal Pe'er, Ralph Pichler, Nishan Portoyan, Meile Posthuma, Adam Roman, Gerhard Runze, Andrew Rutz, Klaus Skafte, Mike Smith, Payal Sobti, Péter Sótér, Michael Stahl, Chris van Bael, Stephanie van Dijck, Robert Werkhoven, Paul Weymouth, Dong Xin, Ester Zabar, Claude Zhang.

日本語訳については、Japan Software Qualifications testing Board メンバおよび以下の日本語翻訳ワーキンググループメンバにより行われた。

日本語翻訳ワーキンググループメンバ：

井芹 洋輝(トヨタ自動車)

須原 秀敏(ベリサーブ)

永田 哲(メタテクノ)

松木 晋祐(ベリサーブ)

真鍋 誠一(センスタイムジャパン)

山口 晋一(慶應義塾大学 SDM 研究所)

0. はじめに

0.1 本シラバスの目的

このシラバスは、ISTQB®認定テスト担当者 AI テスティングのベースとなる。 ISTQB®は本シラバスを次の趣旨で提供する：

1. 各国の委員会に対し、各国語への翻訳および教育機関の認定の目的で提供する。各国の委員会は、本シラバスを各言語の必要性に合わせて調整し、出版事情に合わせてリファレンスを修正することができる
2. 試験委員会に対し、各シラバスの学習目的に合わせ、各国語で試験問題を作成する目的で提供する
3. 教育機関に対し、コースウェアを作成し、適切な教育方法を確定できるようにする目的で提供する
4. 受験志願者に対し、試験準備の目的で提供する（研修コースの一部として、または独立した形態で実施）
5. 国際的なソフトウェアおよびシステムエンジニアリングのコミュニティに対し、ソフトウェアやシステムをテストする技能の向上を目的とする他、書籍や記事を執筆する際の参考として提供する

0.2 The Certified Tester AI Testing

認定テスト担当者 AI テスティングは、AI ベースのシステムのテストやテスト用の AI に携わる人を対象としている。 テスト担当者、テストアナリスト、データアナリスト、テストエンジニア、テストコンサルタント、テストマネージャー、ユーザー受け入れテスト担当者、ソフトウェア開発者などが対象となる。 また、プロジェクトマネージャー、品質マネージャー、ソフトウェア開発マネージャー、ビジネスアナリスト、オペレーションチームメンバー、IT ディレクター、経営コンサルタントなど、AI ベースのシステムのテストやテスト用の AI について基本的な理解を深めたい方にも適した資格である。

認定テスト担当者 AI テスティングの概要 [I03]は、以下の情報を含む別冊のドキュメントである。

- シラバスのビジネス成果
- ビジネス成果と学習目標との関連性を示した表
- シラバスの概要
- 他のシラバスとの関係

0.3 出題可能な学習の目的と認知的な知識のレベル

学習の目的はビジネス成果を支援し、認定テスト担当者 AI テスティングを取得するための試験問題作成を行うために使用する。

受験者は、11 の章のいずれかで言及されているキーワードやコンセプトを認識、記憶、または想起することを求められることがある。 具体的な学習の目的のレベルは、各章の冒頭に示されており、以下のように分類されている。

- K1：記憶
- K2：理解
- K3：適用
- K4：分析

学習の目的に明記されていなくても、章の見出しのすぐ下にキーワードとして記載されている用語はすべて覚えておく（K1）必要がある。

0.4 ハンズオン コンピテンシーのレベル

認定テスト担当者 AI テスティングでは、実践的なスキルとコンピテンシーに焦点を当てた実践的な目的が含まれている。

次のレベルは、ハンズオンの目的に適用される：

- H0：演習のライブデモ、または録画したビデオ
- H1：ガイド付きの演習。トレーナーが実施する一連のステップに生徒が従う
- H2：ヒント付きの演習。 与えられた時間内に演習を解くことができるように、関連するヒントを与えたり、受講生がディスカッションに参加したりする

コンピテンシーは、以下のリストにあるような実践的な演習を行うことで達成される：

- アンダーフィッティングとオーバーフィッティングを実証する（H0）
- ML モデルの作成をサポートするためのデータ準備を行う(H2)
- 訓練データセットとテストデータセットを特定し、ML モデルを作成する（H2）
- 作成された ML モデルを、選択された ML 機能パフォーマンスメトリクスを用いて評価する（H2）
- パーセプトロンの実装を経験する（H1）
- 説明可能性がテスト担当者にどのように使われるかを示すツールを使用する（H2）

- ペアワイズテストを適用して、AI ベースのシステムのテストケースを導き出し、実行する（H2）
- メタモルフィックテストを適用して、与えられたシナリオに対するテストケースを導き出し、実行する（H2）
- 探索的テストを AI ベースのシステムに適用する（H2）
- テストにおいて、AI が使われにくい活動について、例を挙げて議論する（H2）
- AI を使った簡単な欠陥予測システムを実装する（H2）

0.5 認定テスト担当者 AI テスティングの試験

認定テスト担当者 AI テスティングの試験は、このシラバスに基づいて行われる。 試験問題への解答には、このシラバスの複数のセクションに基づいた資料の使用が必要になる場合がある。 シラバスのすべてのセクションは、「はじめに」と「付録」を除いて、試験に出題される。 規格や書籍は参考文献として含まれているが、それらの内容は、規格や書籍からシラバス自体に要約されているものを除き、試験の対象とはならない。

詳細は、認定テスト担当者 AI テスティング"概要"の "試験の構成"の項を参照。

応募条件に関する注記：ISTQB ®Foundation Level 資格は、認定テスト担当者 AI テスティングの試験を受ける前に取得しておく必要がある。

0.6 認定

ISTQB®のメンバー委員会にて、教育コースの教材が本シラバスに従っている教育機関を認定する。教育機関はメンバー委員会または認定を行う機関から認定ガイドラインを入手しなければならない。教育コースがシラバスに従っていると認定されると、教育コースの一部として ISTQB®の試験を実施することができる。

このシラバスの認定ガイドラインは、Processs Management and Compliance Working Group が発表した一般的な認定ガイドラインに従う。

0.7 レベルオブディテール

本シラバスの詳細レベルは国際的に一貫した教育と試験を可能にする。このゴールを達成するために本シラバスは以下のようになっている。

- 全般的な教育内容の目的（認定テスト担当者 AI テスティングの意図について説明）
- 用語のリスト（思い出すことができなければならない用語）
- 各知識領域の学習の目的（達成しなければならない知識レベルの学習の成果について説明）

- 教えるべき主要なコンセプトの説明（承認された文献、および標準を情報源として含む）

シラバスの内容は、AI ベースのシステムをテストするための知識領域全体を記述したものではなく、認定テスト担当者 AI テスティングのトレーニングコースでカバーする詳細レベルを反映している。 特に人工知能（AI）と機械学習の基本的な概念を紹介し、これらの技術をベースにしたシステムをどのようにテストするかに焦点を当てている。

0.8 シラバスの構成

試験可能な内容の章が 11 個ある。各章の一番上の見出しは、章の学習時間を指定している。章より下のレベルでは、時間は指定されていない。認定トレーニングコースの場合、本シラバスでは最低 25.1 時間の講義が要求されており、以下のように 11 の章に分けられる：

- 第 1 章： 105 分 AI の紹介
- 第 2 章： 105 分 AI ベースのシステムの品質特性
- 第 3 章： 145 分 機械学習（ML）- 概要
- 第 4 章: 230 分 ML - データ
- 第 5 章： 120 分 ML 機能パフォーマンスメトリクス
- 第 6 章: 65 分 ML - ニューラルネットワークとテスト
- 第 7 章： 115 分 AI ベースのシステムをテストする 概要
- 第 8 章： 150 分 AI 特有の品質特性のテスト
- 第 9 章： 245 分 AI ベースのシステムのテストのための方法と技法
- 第 10 章 30 分 AI ベースのシステムのテスト環境
- 第 11 章： 195 分 テストに AI を使う

1. AI の紹介 - 105 分

テストキーワード

なし

AI に特化したキーワード

AI as a Service (AIaaS), AI 開発フレームワーク, AI の効果, AI ベースのシステム, 人工知能（AI）, ニューラルネットワーク, ディープラーニング（DL）, ディープニューラルネットワーク, 汎用型 AI, 一般データ保護規則（GDPR）, 機械学習（ML）, 特化型 AI, 学習済みモデル, スーパーAI, 技術的特異点, 転移学習

第 1 章の学習目標

1.1 AI の定義と AI 効果

AI-1.1.1 K2 AI 効果と、それが AI の定義にどのように影響するかを説明する。

1.2 特化型 AI、汎用型 AI、スーパーAI

AI-1.2.1 K2 特化型 AI、汎用型 AI、およびスーパーAI を区別できる。

1.3 AI を使ったシステムと従来のシステム。

AI-1.3.1 K2 AI ベースのシステムと従来のシステムを区別できる。

1.4 AI テクノロジー

AI-1.4.1 K1 AI を実現するために使用されるさまざまな技術を認識することができる。

1.5 AI 開発フレームワーク

AI-1.5.1 K1 一般的な AI 開発フレームワークを特定する。

1.6 AI ベースのシステムのためのハードウエア

AI-1.6.1 K2 AI ベースのシステムを実装するためのハードウェアに利用できる選択肢を比較する

1.7 AI as a Service (AIaaS)

AI-1.7.1 K2 AI as a Service (AIaaS)のコンセプトを説明する。

1.8 学習済みモデル

AI-1.8.1 K2 学習ずみ AI モデルの使用と、それに伴うリスクについて説明する。

1.9 規格・規制・AI

AI-1.9.1 K2 標準規格が AI ベースのシステムにどのように適用されるかを説明する。

1.1 AI の定義と AI 効果

人工知能（AI）という言葉は、1950 年代にさかのぼり、人間を模倣できる「知的な」機械を構築し、プログラミングすることを目的としたものである。今日の定義は大きく進化しており、以下の定義はその概念を捉えている[S01]。

知識や技能を習得し、処理し、適用するための工学的システムの能力。

AI の意味をどのように理解するかは、その人の現在の認識によって異なる。 1970 年代においては、チェスで人間を打ち負かすコンピュータシステムが登場することは未来の話であり、それが AI であると考えられていた。 しかし、コンピュータベースのシステム「Deep Blue」がチェスの世界チャンピオン、ガルリ・カスパロフを破ってから 20 年以上が経過した現在、このシステムで実装されていた「ブルートフォース」アプローチは、多くの人にとって真の人工知能とはみなされていない（つまり、システムはデータから学習せず、自己学習もできなかった）。 同様に、1970～80 年代のエキスパートシステムは、人間の専門知識をルールとして組み込んでおり、専門家がいなくても繰り返し実行することができた。これらは、当時は AI と考えられていたが、現在ではそのようには考えられていない。

AI を構成するものに対する認識の変化は、「AI 効果」として知られている[R01]。 社会における AI の認識が変化すると、その定義も変化する。その結果、現在行われている定義は将来的に変化する可能性が高く、過去の定義とは一致しない可能性がある。

1.2 特化型 AI、汎用型 AI、スーパーAI

高いレベルで、AI は 3 つのカテゴリーに分けることができる。

- 特化型 AI（弱い AI とも呼ばれる）システムは、限られたコンテキストで特定のタスクを実行するようにプログラムされている。 現在、この形式の AI は広く普及している。 例えば、ゲームプレイングシステム、スパムフィルター、テストケースジェネレーター、音声アシスタントなど。
- 汎用型 AI（強い AI とも呼ばれる）システムは、人間と同様の一般的な（広範囲にわたる）認知能力を持っている。 これらの AI ベースのシステムは、人間のように推論し、環境を理解し、それに応じて行動することができる。 2021 年現在、汎用型 AI システムは実現していない。
- スーパーAI システムは、人間の認知を再現することができ（汎用型 AI）、巨大な処理能力、実質的に無制限のメモリ、人間のあらゆる知識へのアクセス（ウェブへのアクセスなど）を利用する。スーパーAI システムは、すぐに人間よりも賢くなると考えられている。AI ベースのシステムが汎用型 AI からスーパーAI に移行する時点は、一般的に技術的特異点として知られている[B01]。

1.3 AI ベースのシステムと従来のシステム

一般的なコンピュータシステムでは、ソフトウェアは人間が if-then-else やループなどの構文を含む命令型言語を使ってプログラミングする。人間は、システムがどのように入力を出力に変換するかを比較的容易に理解することができる。機械学習（ML）を利用した AI ベースのシステムでは、データのパターンを利用して、新しいデータに対して将来どのように反応すべきかをシステムが判断する（ML についての詳細な説明は第 3 章を参照）。 例えば、猫の画像を識別するために開発された AI ベースの画像処理装置は、猫が含まれていることが知られている画像のセットで学習される。AI は、データに含まれるどのようなパターンや特徴が猫の識別に利用できるかを自分で判断する。そして、そのパターンやルールを新しい画像に適用して、その画像に猫が写っているかどうかを判断する。多くの AI ベースのシステムでは、このような予測作成手順は、人間にとって理解しにくいものとなっている（2.7 項参照）。

実際には、AI ベースのシステムはさまざまな技術によって実現される（1.4 項参照）。また、「AI 効果」（1.1 項参照）によって、現在、何が AI ベースのシステムとみなされ、何が従来のシステムとみなされるかが決まることもある。

1.4 AI 技術

AI は、以下のような様々な技術（詳細は[B02]を参照）を用いて実装することができる。

- ファジーロジック
- 検索アルゴリズム
- 推論技術
  - ルールエンジン
  - 演繹的分類法
  - 事例ベース推論
  - 手続き型推論
- 機械学習技術
  - ニューラルネットワーク
  - ベイズモデル
  - 決定木
  - ランダムフォレスト
  - 線形回帰
  - ロジスティック回帰
  - クラスタリング・アルゴリズム

- 遺伝的アルゴリズム
- サポートベクターマシン（SVM）

AI を使ったシステムは、一般的にこれらの技術を 1 つ以上実装している。

1.5 AI 開発フレームワーク

多くの AI 開発フレームワークがあるが、中には特定のドメインに特化したものもある。これらのフレームワークは、データの準備、アルゴリズムの選択、中央演算処理装置（CPUs）、グラフィカル・プロセッシング・ユニット（GPUs）、クラウド・テンソル・プロセッシング・ユニット（TPUs）などのさまざまなプロセッサ上で動作するモデルのコンパイルなど、さまざまな活動をサポートする。また、特定のフレームワークを選択する際には、実装に使用するプログラミング言語や使いやすさなどの特定の側面に依存する場合もある。代表的なフレームワークとして、以下のものがある（2021 年 4 月現在）。

- Apache MxNet：Amazon が Amazon Web Services（AWS）で使用しているディープラーニングのオープンソースフレームワーク[R02]。
- CNTK：Microsoft Cognitive Toolkit（CNTK）は、オープンソースの深層学習ツールキット[R03]。
- IBM Watson Studio：AI ソリューションの開発を支援するツール群[R04]。
- Keras：Python 言語で書かれたハイレベルなオープンソースの API で、TensorFlow や CNTK の上で動作させることができる[R06]。
- PyTorch：Facebook 社が運営するオープンソースの ML ライブラリで、画像処理や自然言語処理（NLP）を応用したアプリに利用されている。Python と C++の両方のインターフェースがサポートされている[R07]。
- Scikit-learn：プログラミング言語 Python のためのオープンソースの ML ライブラリ[R08]。
- TensorFlow：スケーラブルな機械学習のためのデータフローグラフに基づくオープンソースの ML フレームワークで、Google が提供している[R05].

なお、これらの開発フレームワークは、時に融合し、時に新しいフレームワークに置き換わるなど、常に進化している。

1.6 AI ベースのシステムのためのハードウェア

ML モデルの学習（第 3 章参照）やモデルの実装には、さまざまなハードウェアが使用されている。例えば、音声認識を行うモデルは、ローエンドのスマートフォンでも動作するが、モデルの学習にはクラウドコンピューティングの能力が必要となる場合がある。ホストデバイスがインターネットに接続されていない場合、クラウドでモデルを学習してからホストデバイスにデプロイするというアプローチが一般的である。

ML では、次のような特性を持つハードウェアが有効である。

- 低精度の演算：計算に使用するビット数が少ない（例えば、ML では通常必要な 32 ビットではなく 8 ビット）。
- 大規模なデータ構造を扱う能力（例：行列の乗算をサポートするため）。
- 超並列（コンカレント）処理。

汎用 CPU は、ML アプリケーションには必要のない複雑な演算をサポートし、数コアしか備えていない。そのため、数千のコアを持ち、超並列かつ比較的単純な画像処理を行うことを目的とした GPU と比較すると、ML モデルの学習・実行には効率的でないアーキテクチャとなっている。その結果、CPU の方がクロックスピードが速いにもかかわらず、GPU は ML アプリケーションにおいて CPU よりも優れた性能を発揮する。小規模な ML 作業では、一般的に GPU が最適な選択肢となる。

ハードウェアの中には、目的に応じて作られた ASIC（Application-Specific Integrated Circuits）や SoC（Systems on a Chip）など、AI に特化したものがある。これらの AI に特化したソリューションは、マルチコア、特殊なデータ管理、インメモリ処理の機能などを備えている。これらはエッジコンピューティングに最も適しており、ML モデルのトレーニングはクラウド上で行われる。

現在（2021 年 4 月時点）、特定の AI アーキテクチャを持つハードウェアが開発されている。これには、従来のフォン・ノイマン・アーキテクチャではなく、脳のニューロンを大雑把に模倣したアーキテクチャを採用したニューロモーフィック・プロセッサー[B03]が含まれる。

AI ハードウェアプロバイダーとそのプロセッサーの例は以下の通り（2021 年 4 月現在）。

- NVIDIA：様々な GPU や、「Volta」[R09]などの AI 専用プロセッサを提供している。
- Google：彼らは、学習と推論の両方のために、アプリケーションに特化した集積回路を開発した。 「Google TPU（Cloud Tensor Processing Units）」[R10]は、Google Cloud 上でユーザーがアクセスできるのに対し、「Edge TPU」[R11]は、個々のデバイス上で AI を実行するために設計された専用の ASIC である。
- インテル：ディープラーニング（学習と推論の両方）用の「Nervana ニューラルネットワークプロセッサ」[R12]と、コンピュータビジョンとニューラルネットワークアプリケーションにおける推論用の「Movidius Myriad ビジョンプロセッシングユニット」を提供している。
- Mobileye：複雑で計算量の多い画像処理をサポートする SoC デバイス「EyeQ ファミリー」[R13]を製造している。これらのデバイスは、自動車での使用のために低消費電力である。
- アップル：iPhone に搭載されているオンデバイス AI 用の「Bionic チップ」を製造している[B04]。
- ファーウェイ：同社のスマートフォン用チップ「Kirin 970」には、AI 用のニューラルネットワーク処理が組み込まれている[B05]。

1.7 AI as a Service (AIaaS)

ML モデルなどの AI コンポーネントは、組織内で作成することも、サードパーティからダウンロードすることも、Web 上のサービス（AIaaS）として利用することも可能である。また、AI 機能の一部をシステム内で提供し、一部をサービスとして提供するというハイブリッドなアプローチも可能である。

ML をサービスとして利用する場合、Web 上で ML モデルへのアクセスを提供し、データの準備や保管、モデルの訓練、評価、チューニング、テスト、デプロイメントなどのサポートも提供することができる。

第三者のプロバイダー（AWS、Microsoft など）は、顔認識や音声認識などの特定の AI サービスを提供している。これにより、個人や組織が独自の AI サービスを構築するためのリソースや専門知識が不十分な場合でも、クラウドサービスを利用して AI を導入することができる。また、サードパーティサービスの一部として提供される ML モデルは、最近 AI 市場に参入した人たちなど、多くのステークホルダーが容易に入手できるものよりも、より大規模で多様な学習データセットで学習されている可能性が高い。

1.7.1 AI as a Service の契約

これらの AI サービスは、通常、非 AI のクラウドベースの SaaS（Software as a Service）と同様の契約で提供される。AIaaS の契約には、一般的に、可用性とセキュリティのコミットメントを定義するサービスレベル契約（SLA）が含まれる。このような SLA は、通常、サービスの稼働率（例：99.99%の稼働率）と不具合修正のための応答時間をカバーしているが、ML の機能的なパフォーマンス指標（例：精度）を同様に定義することはほとんどない（第 5 章参照）。 AIaaS は多くの場合、サブスクリプションベースで支払われ、契約した稼働率や応答時間が満たされない場合、サービスプロバイダーは一般的に将来のサービスに対するクレジットを提供する。このようなクレジットを除き、ほとんどの AIaaS の契約は（支払った料金以外の）責任を限定しているため、AIaaS に依存する AI ベースのシステムは通常、サービスの損失がそれほどダメージにならない、比較的リスクの低いアプリケーションに限定される。

サービスには、多くの場合、受け入れ期間の代わりに最初の無料試用期間が設けられている。この期間中に、AIaaS の利用者は、提供されたサービスが必要な機能や性能（例：正確さ）の点でニーズを満たしているかどうかをテストすることが期待される。これは一般的に、提供されるサービスの透明性の欠如をカバーするために必要である（7.5 項参照）。

1.7.2 AIaaS の例

以下は、AIaaS の例である（2021 年 4 月現在）。

- IBM Watson Assistant：月間アクティブユーザー数に応じて価格が決まる AI チャットボット。
- Google Cloud AI and ML Products：フォームパーサーやドキュメント OCR を含むドキュメントベースの AI を提供している。価格は、処理のために送信されるページ数に基づく。
- Amazon CodeGuru：ML が Java コードのレビューを行い、開発者にコード品質向上のための推奨事項を提供する。価格は解析したソースコードの行数に応じて決まる。

- Microsoft Azure Cognitive Search：AI クラウド検索を提供する。価格は検索単位（使用するストレージやスループットで定義）に基づく。

1.8 学習済みモデル

1.8.1 学習済みモデルの紹介

ML モデルの学習にはコストがかかる（第 3 章参照）。まず、データを準備し、次にモデルを訓練する必要がある。前者は大量の人的資源を消費し、後者は大量のコンピューティング資源を消費する。多くの組織は、これらのリソースを利用できない。

安価でより効果的な方法は、事前にトレーニングされたモデルを使用することである。これは、必要なモデルと同様の機能を提供し、学習済みモデルの機能を拡張および／または特化させた新しいモデルを作成するための基盤として使用される。このようなモデルは、ニューラルネットワークやランダムフォレストなど、限られた技術でしか利用できない。

画像分類器が必要な場合は、1,400 万枚以上の画像を 1,000 以上のカテゴリに分類した ImageNet データセットを使って学習することができる。これにより、成功の保証がないまま多大なリソースを消費するリスクを軽減できる。 また、このデータセットで既に学習された既存のモデルを再利用することも可能である。このように学習済みモデルを使用することで、学習コストを削減し、動作しないリスクをほぼ排除することができる。

学習済みモデルをそのまま使用する場合、AI ベースのシステムに単純に組み込むことも、サービスとして利用することもできる（1.7 節参照）。

1.8.2 転移学習

また、学習済みモデルを修正して、別の要件を実行することも可能である。これは転移学習と呼ばれ、ディープニューラルネットワークで使用される。ニューラルネットワークの初期層（第 6 章参照）は通常、非常に基本的なタスクを実行し（例：画像分類器で直線と曲線の違いを識別する）、後期層はより専門的なタスクを実行する（例：建物の建築タイプを区別する）。この例では、画像分類器の後期層以外を再利用することで、初期層の学習が不要になる。その後、新しい分類器に特有の要件を満たすために、後期層を再学習する。実際には、学習済みモデルは、新しい問題固有のデータで追加学習することにより、微調整することができる。

このアプローチの有効性は、元のモデルが実行する機能と、新しいモデルが要求する機能の類似性に大きく依存する。例えば、猫の種類を識別する画像分類器を犬の種類を識別するように変更することは、人のアクセントを識別するように変更することよりもはるかに効果的である。

多くの学習済みモデルが、特に学術研究者から提供されている。そのような学習済みモデルの例としては、画像分類用の Inception、VGG、AlexNet、MobileNet などの ImageNet モデル[R14]や、Google の BERT[R15]などの学習済みの NLP モデルがある。

1.8.3 学習済みモデルと転移学習のリスク

学習済みモデルや転移学習を利用することはともに、AI ベースのシステムを構築するための一般的なアプローチであるが、それらには以下のようないくつかのリスクが存在する。

- 学習済みモデルは、内部で生成されたモデルに比べて透明性に欠ける場合がある。
- 学習済みモデルが実行する機能と、要求される機能との類似度合いが不十分な場合がある。また、この違いがデータサイエンティストに理解されていない場合もある。
- 初期開発時の事前学習済みモデルのデータ準備手順（セクション 4.1 参照）と、このモデルを新システムで使用する際のデータ準備手順が異なる場合、 結果として機能パフォーマンスに影響を与える可能性がある。
- 学習済みモデルの欠点として、それを再利用する人に文書化されないまま継承される可能性が高いというものがある。例えば、モデルの学習に使用したデータに関する文書がない場合、継承されたバイアス（セクション 2.4 参照）は明らかにならないかもしれない。また、学習済みモデルが広く使用されていない場合、未知の（あるいは文書化されていない）欠陥がより多く存在する可能性があり、このリスクを軽減するために、より厳密なテストが必要となる場合がある。
- 転移学習によって作成されたモデルは、そのモデルの基となった学習済みモデルと同じ脆弱性（例えば、9.1.1 で説明したような敵対的攻撃）に対して敏感に反応する可能性が高い。また、AI ベースのシステムに特定の学習済みモデルが含まれている（または特定の学習済みモデルに基づいている）ことが知られている場合、そのシステムに関連する脆弱性は、潜在的な攻撃者によってすでに知られている可能性がある。

なお、上記のリスクのうちいくつかは、学習済みモデルの詳細なドキュメントを用意することで、より簡単に軽減することができる（セクション 7.5 参照）。

1.9 規格・規制・AI

IEC と ISO の情報技術に関する合同技術委員会（ISO/IEC JTC1）は、AI に対して貢献する国際規格を作成している。例えば、AI に関する小委員会（ISO/IEC JTC1/SC42）は、2017 年に設置された。また、ソフトウェアやシステム工学を扱う ISO/IEC JTC1/SC7 では、「AI ベースのシステムのテスト」に関するテクニカルレポートを発表している[S01]。

また、AI に関する規格は、地域レベル（欧州規格など）や国レベルでも発表されている。

EU 全体の一般データ保護規則（GDPR）は 2018 年 5 月に施行され、個人データと自動化された意思決定に関するデータ管理者の義務を定めている[B06]。その中には、潜在的な差別の回避を含む AI システムの機能パフォーマンスの評価と改善、および自動化された意思決定に従わない個人の権利を確保するための要件が含まれている。テストの観点から見た GDPR の最も重要な点は、個人データ（予測を含む）が正確でなければならないということである。これは、システムが行う予測の一つ一つが正確でなければならないという意味ではなく、システムが使用される目的に対して十分な精度を持っていなければならないということを指す。

また、ドイツの国家標準化団体（DIN）は、AI 品質メタモデルを開発した（[S02]、[S03]）。

AI に関する基準は、業界団体からも発表されている。例えば、IEEE（Institute of Electrical and Electronics Engineers）では、倫理と AI に関するさまざまな規格を策定している（The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems）。これらの規格の多くは、本稿執筆時点ではまだ開発中である。

AI が安全関連システムに使用される場合、自動車システムに関する ISO 26262 [S04]や ISO/PAS 21448 (SOTIF) [S05]などの関連する規制基準が適用される。このような規制規格は通常、政府機関によって義務付けられており、国によっては搭載されているソフトウェアが ISO 26262 に準拠していなければ、自動車を販売することが違法となる場合もある。規格はあくまでも任意の文書であり、その使用は通常、法律や契約によってのみ義務化される。しかし、多くのユーザーは、作成者の専門知識を活用し、より質の高い製品を作るために標準規格を利用している。

2. AI ベースのシステムの品質特性 - 105 分

キーワード

なし

AI に特化したキーワード

適応性、アルゴリズムバイアス、自律性、バイアス、進化、説明可能性、説明可能な AI（XAI）、柔軟性、不適切なバイアス、解釈可能性、ML システム、機械学習、報酬ハッキング、堅牢性、サンプリングバイアス、自己学習型システム、副作用、透明性

第 2 章の学習目標

2.1 柔軟性と適応性

AI-2.1.1 K2 AI ベースのシステムの特徴として、柔軟性と適応性の重要性を説明する。

2.2 自律性

AI-2.2.1 K2 自律性と AI ベースのシステムの関係を説明する。

2.3 進化

AI-2.3.1 K2 AI ベースのシステムの進化を管理することの重要性を説明する。

2.4 バイアス

AI-2.4.1 K2 AI ベースのシステムに見られるさまざまな原因とバイアスの種類を説明する。

2.5 倫理

AI-2.5.1 K2 AI ベースのシステムの開発、デプロイ、使用において尊重されるべき倫理原則について議論する。

2.6 副作用と報酬のハッキング

AI-2.6.1 K2 AI ベースのシステムにおける副作用の発生と報酬ハッキングについて説明する。

2.7 透明性、解釈可能性、説明可能性

AI-2.7.1 K2 透明性、解釈可能性、説明可能性が AI ベースのシステムにどのように適用されるかを説明する。

2.8 安全性と AI

AI-2.8.1 K1 AI を用いたシステムを安全関連のアプリケーションで使用することを困難にする特性を想起する。

2.1 柔軟性と適応性

柔軟性と適応性は、密接に関連する品質特性である。このシラバスでは、「柔軟性」とは、当初のシステム要件に含まれていなかった状況でもシステムを使用できる能力であり、「適応性」とは、ハードウェアの違いや運用環境の変化など、新たな状況に応じてシステムを容易に変更できる能力であると考える。

柔軟性と適応性はともに以下のような場合に役立つ：

- システムの導入時には運用環境が完全にはわからない。
- システムは新しい運用環境に対応することが期待される。
- システムは新しい状況に適応することが期待される。
- システムはいつその動作を変えるべきかを判断しなければならない。

自己学習型の AI ベースのシステムは、上記の特性のすべてを示すことが期待される。結果として、それらは適応性があり、柔軟に対応できる可能性を持っている必要がある。

AI ベースのシステムの柔軟性と適応性の要件には、システムが適応することが期待される環境の変化の詳細を含める必要がある。また、これらの要件には、システムが適応するために使用できる時間とリソースに関する制約を明記する必要がある（例：新しいタイプの物体を認識するための適応にどれくらいの時間がかかるか）。

2.2 自律性

自律性を定義する際に重要なのは、完全に自律したシステムは人間の監視や制御から完全に独立しているということをまず認識することである。しかし、実際には、完全な自律性が求められることはあまりない。例えば、一般に「自律型」と呼ばれている完全自動運転車は、公式には「完全自動運転化」と分類されている[B07]。

多くの人は、自律システムを「スマート」または「インテリジェント」であると考えており、それは、特定の機能を実行するために AI ベースのコンポーネントを含むことを示唆している。例えば、状況認識を必要とする自律走行車は、複数のセンサーと画像処理を用いて、車両の直近の環境に関する情報を収集する。この機能を実現するためには、機械学習、特に深層学習（6.1 項参照）が最も効果的なアプローチであることがわかっている。自律システムには、意思決定や制御の機能も含まれる。これらはいずれも、AI ベースのコンポーネントを使って効果的に実行できる。

一部の AI ベースのシステムは自律的であると考えられているが、これらはすべての AI ベースのシステムに当てはまるわけではない。このシラバスでは、自律性とは、システムが長時間にわたって人間の監視や制御から独立して動作する能力であると考える。これはつまり、自律性を持つシステムが、仕様化されテストされる必要のあるシステム自身の特性を識別できることを意味する。例えば、自律システムが人間の介入なしに満足のいく動作をすると期待される時間の長さを知る必要がある。また、自律システムが人間の制御者に制御を返さなければならない事象を特定することも重要である。

2.3 進化

このシラバスでは、進化とは、変化する外部制約に対応してシステムが自己を改善する能力だと考えている。AI システムの中には自己学習型と言えるものがあり、自己学習型の AI ベースのシステムを成功させるには、このような進化の形を取り入れる必要がある。

AI ベースのシステムは、進化する環境の中で運用されることが多い。他の IT システムと同様に、AI ベースのシステムは、その運用環境の変化に対応できる柔軟性と適応性が必要である。

自己学習型の AI ベースのシステムでは、一般的に以下のふたつの形態の変化を管理する必要がある。

- 変化の一つの形は、システムが自らの決定や環境との相互作用から学習することである。
- もう一つの変化の形は、運用環境に加えられた変化からシステムが学習する場合である

どちらの場合も、システムは理想的にはその有効性と効率性を向上させるために進化する。しかし、この進化は、システムが望ましくない特性を持つようにならないように制約を受けなければならない。どのような進化であっても、元のシステム要件と制約を満たし続けなければならない。これらが欠けている場合は、進化が制限内にとどまり、常に人間の価値観に沿ったものになるようにシステムを管理しなければならない。2.6 節では、自己学習型 AI ベースのシステムにおける副作用と報酬ハッキングの影響に関する例を示している。

2.4 バイアス

AI ベースのシステムにおけるバイアスとは、システムが提供する出力と、特定のグループへのひいき目無しの「公正な出力」と考えられるものとの間の距離を統計的に測定したものである。不適切なバイアスは、性別、人種、民族、性的指向、所得水準、年齢などの属性に関連している。AI を用いたシステムにおいて、不適切なバイアスが発生するケースは、例えば、銀行の融資を提案するシステム、採用システム、司法監視システムなどで報告されている。

バイアスは、さまざまな AI ベースのシステムに入り込む可能性がある。例えば、エキスパートシステムが採用するルールに専門家のバイアスが組み込まれることを防ぐことは困難である。しかし、ML システムの普及は、バイアスに関する議論の多くが ML システムのコンテキストで行われていることを意味する。

ML システムは、収集したデータを使用したアルゴリズムを用いて判断や予測を行うものであり、これらの 2 つの要素が結果にバイアスをもたらす可能性がある。

- アルゴリズムのバイアスは、学習アルゴリズムが正しく設定されていない場合に発生する。例えば、あるデータを他のデータと比較して過大評価している場合などである。このバイアスのソースは、ML アルゴリズムのハイパーパラメータの調整により制御することができる（セクション 3.2 参照）。
- サンプリングバイアスは、学習データが ML の適用先のデータ空間を十分に代表していない場合に発生する。

不適切なバイアスは、サンプリングバイアスに起因することが多いが、アルゴリズムバイアスに起因する場合もある。

2.5 倫理

倫理とは、ケンブリッジ辞典では次のように定義されている。

行動を制御するための受け入れられた信念のシステム、特に道徳に基づくシステム。

AI を活用して機能を強化したシステムは、人々の生活に大きくプラスの影響を及ぼしている。これらのシステムが普及するにつれ、倫理的に正しい使い方がされているかどうかが懸念されるようになった。

何が倫理的であると考えられるかは、時間の経過とともに変化し、また地域や文化によっても変わる可能性がある。AI ベースのシステムをある場所から別の場所に展開する際には、ステークホルダーの価値観の違いを考慮するよう注意しなければならない。

AI の倫理に関する国や国際的な政策は、多くの国や地域で見られる。経済協力開発機構（OECD）は、AI の責任ある開発のために各国政府が合意した初の国際基準である「AI のための原則」を 2019 年に発行した[B08]。この原則は、発行時に 42 カ国で採択され、欧州委員会も支持している。それらには、実践的な政策提言に加えて、「信頼できる AI の責任ある管理」のための価値観に基づく原則が含まれている。これらは次のようにまとめられている。

- AI は、包括的な成長、持続可能な開発、幸福を促進することで、人々と地球に利益をもたらすものでなければならない。
- AI システムは、法の支配、人権、民主主義の価値、多様性を尊重し、公正な社会を確保するための適切なセーフガードを含むべきである。
- 人々が結果を理解し、それに異議を唱えることができるよう、AI に関する透明性を確保する必要がある。
- AI システムは、そのライフサイクルを通じて、堅牢で安全かつ安心な方法で機能しなければならず、リスクは継続的に評価されなければならない。
- AI システムを開発、展開、運用する組織や個人は、説明責任を果たすべきである。

2.6 副作用と報酬ハッキング

副作用や報酬ハッキングは、AI ベースのシステムがその目標を達成しようとするときに、予期しない、あるいは有害な結果を生み出す可能性がある[B09]。

AI ベースのシステムの設計者が、「環境内の特定のタスクを 達成することに焦点を当て、（潜在的に非常に大きな）環境の他の側面を無視し、したがって、変更すると実際に有害であるかもしれない環境変数に対して暗黙のうちに無関心を示す」 目標を指定した場合，負の副作用が生じる可能性がある[B09]。例えば、「できるだけ低燃費で安全な方法」で目的地まで移動することを目標とした自動運転車は、目標を達成しても、時間がかかりすぎて乗客が非常に困るという副作用があるかもしれない。

報酬ハッキングは、AI ベースのシステムが、「設計者の意図を曲解する」ような「巧妙な」または「簡単な」解決策を用いて指定されたゴールを達成することで生じる可能性がある。これは、効果的にゴールの達成をハックするようなことである。報酬ハッキングの例としてよく使われるのが、AI を使ったシステムがアーケードゲームのプレイを自律学習するというものだ。このゲームでは、「最高得点」を達成することが目標とされているが、そのために AI はゲームをプレイするのではなく、最高得点を記録しているデータレコードをハッキングする。

2.7 透明性、解釈可能性、説明可能性

AI ベースのシステムは、通常、ユーザーがそれらのシステムを信頼する必要がある分野で適用される。これは、安全上の理由だけでなく、プライバシーが必要な場合や、人生を変える可能性のある予測や判断を提供する可能性がある場合などでも同様である

ほとんどのユーザーは AI ベースのシステムを「ブラックボックス」として提示され、これらのシステムがどのようにして結果を導き出すのかをほとんど意識していない。場合によっては、この無知は、システムを構築したデータサイエンティストにも当てはまるかもしれない。場合によっては、ユーザーは自分が AI ベースのシステムと対話していることにさえ気づかないこともある。

AI ベースのシステムには固有の複雑さがあることから、「Explainable AI」（XAI）という分野が生まれた。XAI の目的は、AI ベースのシステムがどのようにして結果を導き出すのかをユーザーが理解できるようにすることで、ユーザーの AI に対する信頼性を高めることにある。

英国王立協会[B10]によると、XAI を望む理由はいくつかある。

- ユーザーへのシステムの信頼感の提供
- 偏見からの保護
- 規制基準や政策要求への適合
- システム設計の改善
- リスク、堅牢性、脆弱性の評価
- システムからの出力の理解と検証
- 自律性、行為主体性（ユーザーに力を感じさせること）、社会的価値への対応

このことから、ステークホルダーの視点から見た、AI を使ったシステムの基本的な望ましい XAI の特性は、以下の 3 つになる（8.6 項も参照）。

- 透明性：モデルを生成するために使用されたアルゴリズムや学習データの特定が容易であること。
- 解釈可能性: これは、ユーザーを含む様々なステークホルダーによる AI 技術の理解可能性と考えられる。

- 説明可能性: AI を使ったシステムがどのようにして特定の結果を導き出したのか、をユーザーが判断しやすいかどうか。

2.8 安全性と AI

このシラバスでは、安全性とは、AI ベースのシステムが人や財産、環境に害を及ぼさないと期待するものと考えている。AI ベースのシステムは、安全性に影響を与える決定を下すために使用されることがある。例えば、医療、製造、防衛、セキュリティ、輸送などの分野で稼働する AI ベースのシステムは、安全性に影響を与える可能性がある。

AI を用いたシステムの安全性（人間に危害を加えないなど）の確保を難しくしている特徴として、以下のようなものがある：

- 複雑性
- 非決定性
- 確率論的性質
- 自己学習
- 透明性、解釈可能性、説明可能性の欠如
- 堅牢性の欠如

これらの特性のいくつかをテストするための課題は、第 8 章で取り上げる。

3. 機械学習（ML） - 概要 - 145 分

キーワード

なし

AI に特化したキーワード

アソシエーション分析、分類、クラスタリング、データ準備、ML アルゴリズム、ML フレームワーク、ML 機能性能基準、ML モデル、ML 訓練データ、ML ワークフロー、モデル評価、モデルチューニング、外れ値、オーバーフィッティング、回帰、強化学習、教師あり学習、アンダーフィッティング、教師なし学習

第 3 章の学習目標

3.1 ML の種類

AI-3.1.1 K2 教師あり学習の一部である分類と回帰について説明する。

AI-3.1.2 K2 教師なし学習の一部であるクラスタリングとアソシエーション分析について説明する

AI-3.1.3 K2 強化学習について説明する。

3.2 ML ワークフロー

AI-3.2.1 K2 ML システムを構築するためのワークフローを要約する。

3.3 ML の形式を選択する

AI-3.3.1 K3 プロジェクトのシナリオが与えられたとき、（分類、回帰、クラスタリング、アソシエーション分析、強化学習の中から）適切な ML の形式を特定できる。

3.4 ML アルゴリズムの選択に関わる要素

AI-3.4.1 K2 ML アルゴリズムの選択に関わる要因を説明する。

3.5 オーバーフィッティングとアンダーフィッティング

AI-3.5.1 K2 アンダーフィッティングとオーバーフィッティングの概念を要約する。

HO-3.5. 1H0 アンダーフィッティングとオーバーフィッティングを実演する。

3.1 ML の種類

ML のアルゴリズムは以下のように分類できる。

- 教師あり学習
- 教師なし学習
- 強化学習

3.1.1 教師あり学習

この種の学習では、アルゴリズムは学習段階でラベル付きデータから ML モデルを作成する。 ラベル付きデータは、通常、入力のペア（例えば犬の画像と「犬」というラベル）で構成されており、アルゴリズムは、学習中に入力データ（例えば犬の画像）と出力ラベルの間の関係（例えば「犬」と「猫」）を推論するために使用される。 ML モデルのテスト段階では、新しい未使用のデータセットを、出力を予測するために訓練済のモデルに適用する。モデルの出力精度が満足できるレベルに達した時点でモデルはデプロイされる。

教師あり学習によって解決される問題は、次の 2 つのカテゴリーに分けられる。

- 分類：入力をあらかじめ定義されたいくつかのクラスのいずれかに分類することを問題が求める場合、分類が使用される。 顔認識や画像内の物体検出は、分類を使用する問題の例である。
- 回帰：ML モデルによる数値出力の予測を問題が求める場合、回帰が使用される。例えば、生活習慣の入力による年齢予測や、株式の将来価格の予測を行う問題が回帰を利用する例である。

なお、ML 問題のコンテキストで使われる回帰（regression）という用語は、[I01]のような他の ISTQB® シラバスで、ソフトウェアの変更が変更関連の欠陥を引き起こす問題を表現するために使われる regression とは異なる。

3.1.2 教師なし学習

この種の学習では、アルゴリズムは学習段階でラベルのないデータから ML モデルを作成する。 ラベルのないデータは、アルゴリズムが学習中に入力データのパターンを推測するために使用され、共通点に基づいて入力を異なるクラスに割り当てる。 テスト段階では、学習したモデルを新たな未入力データに適用し、入力データがどのクラスに割り当てられるべきかを予測する。 モデルは、出力精度が満足のいくレベルに達した時点でデプロイされる。

教師なし学習で解決される問題は、2 つのカテゴリーに分けられる。

- クラスタリング：入力されたデータの類似性を識別し、共通の特徴や属性に基づいてグループ化することを問題が求める場合に使用される。例えば、クラスタリングはマーケティングのために顧客を異なるカテゴリに分類するために使用される。

- アソシエーション分析：これは、データ属性間から興味深い関係や依存関係を特定することを問題が求める場合に使用される。例えば、製品推奨システムでは、顧客の購買行動に基づいて関連性を特定することができる。

3.1.3 強化学習

強化学習とは、システム（知的エージェント）が環境とのやりとりから反復的に学習することで、経験から学習するアプローチである。 強化学習では、訓練データを使用しない。エージェントは、正しい判断をした場合には報酬を獲得し、間違った判断をした場合にはペナルティを受ける。

環境の設定、エージェントが求められた目標を達成するための適切な戦略選択、報酬関数の設計は、強化学習を実装する際の重要な課題である。強化学習を用いたアプリケーションの例としては、ロボット工学、自動走行車、チャットボットがある。

3.2 ML ワークフロー

機械学習のワークフローには次の活動がある：

目的の理解

デプロイする ML モデルの目的を理解し、ステークホルダと合意しながら、ビジネスの優先順位と整合性を確保する必要がある。 開発したモデルに対する受け入れ基準（ML 機能パフォーマンスメトリクスを含む-第 5 章参照）を定義する必要がある。

フレームワークの選択

適切な AI 開発フレームワークは、目的、受け入れ基準、ビジネスの優先順位に基づいて選択する必要がある（1.5 項参照）。

アルゴリズムの選択と構築

ML のアルゴリズムは、目的、受け入れ基準、利用可能なデータなど、様々な要素に基づいて選択される（3.4 項参照）。アルゴリズムは手動でコーディングされる場合もあるが，多くの場合，事前に開発されたコードのライブラリから選択される。 そして必要に応じて、モデルの訓練の準備のためにアルゴリズムをコンパイルする。

データの準備

データの準備（セクション 4.1 参照）には、データの取得、データの前処理、特徴量エンジニアリングが含まれる。これらの作業と並行して、探索的データ解析（EDA）を行うこともある。

アルゴリズムとモデルが使用するデータは、目的に応じたものであり、図 1 に示す「モデルの生成とテスト」のすべての活動で使用される。 例えば、システムがリアルタイム取引システムであれば、データは取引市場から取得する。

モデルの訓練、チューニング、テストに使用するデータは、モデルで使用される運用データを代表するものでなければならない。 場合によっては、モデルの初期訓練のために事前に収集した

データセットを使用することがある（例えば、Kaggle datasets [R16]を参照）。そうでない場合は、生データは通常、何らかの前処理と特徴エンジニアリングを必要とする。

データと自動化されたデータ準備手順はテストを行う必要がある。入力データのテストの詳細については、セクション 7.2.1 を参照すること。

モデルの訓練

選択した ML アルゴリズムは、モデルを訓練するために訓練データを使用する。

ニューラルネットワークを生成するような一部のアルゴリズムは、訓練データセットを何度も読み込む。訓練データセットによる訓練の各反復をエポックと呼ぶ。

アルゴリズムにはモデルの構造（例えばニューラルネットワークの層の数や決定木の深さ）を定義するパラメータが渡される。これらのパラメータは、モデルのハイパーパラメータとして知られている。

アルゴリズムには訓練を制御するパラメータ（例えばニューラルネットワークの訓練に使用するエポック数）も渡される。これらのパラメータは、アルゴリズムのハイパーパラメータとして知られている。

モデルの評価

モデルは、検証データセットを用いて、合意された ML 機能パフォーマンスメトリクスに対して評価され、その結果はモデルの改善（チューニング）に利用される。 モデルの評価とチューニングは、明快な文書を使って管理された条件下で慎重に行う、科学実験のようなものである必要がある。 実際には、異なるアルゴリズム（例えばランダムフォレスト、SVM、ニューラルネットワーク）を用いて複数のモデルを作成・訓練し、評価とチューニングの結果に基づいて最適なモデルを選択するのが一般的である。

モデルのチューニング

合意された ML 機能パフォーマンスメトリクスに照らしてモデルを評価した結果は、データに適合するようにモデルの設定を調整し、それによって性能を向上させるために使用される。 モデルのチューニングは、ハイパーパラメータのチューニングによって行われる。このチューニングでは、訓練活動が変更されたり（例えば，訓練ステップ数の変更や，訓練に使用されるデータ量の変更）、モデルの属性（例えば，ニューラルネットワークのニューロン数や決定木の深さ）が更新されたりする

訓練、評価、チューニングの 3 つの活動は、図 1 に示すように、モデル生成を構成するものと考えることができる。

モデルのテスト

モデルが生成されたら（すなわち，訓練，評価，チューニングが行われたら）、独立したテストデー タセットを使ってテストを行い、合意された ML 機能パフォーマンスメトリクスが満たされていることを確認する必要がある（7.2.2 項参照）。テストで得られた機能パフォーマンス測定

量は、評価で得られたものとも比較され、独立データを用いたモデルのパフォーマンスが評価時よりも著しく低い場合には、別のモデルを選択しなければならない場合がある。

機能パフォーマンステストに加えて、モデルの訓練時間や、予測にかかる時間とリソース使用量などの非機能テストも実行する必要がある。通常、これらのテストはデータエンジニア／サイエンティストによって実行されるが、ドメインについての十分な知識と関連リソースへのアクセス権を持つテスト担当者も実行できる。

モデルのデプロイ

図 1 に示すように、モデルの開発が完了すると、チューニングされたモデルは、関連するデータパイプラインを含む関連リソースとともに、デプロイのために再設計される必要がある。これは通常、フレームワークによって実現される。 対象には組み込みシステムやクラウドが含まれる場合があり、web API を介してそのモデルにアクセスできる。

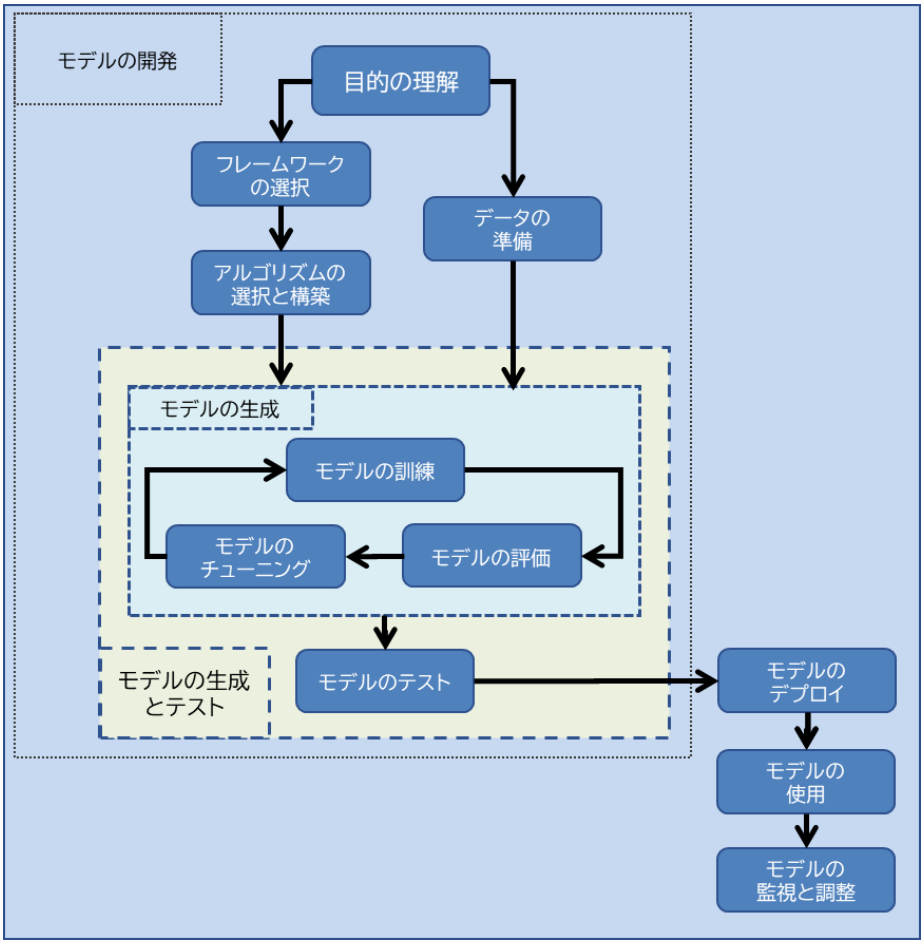

図 1：ML ワークフロー

モデルの使用

デプロイされたモデルは、通常、より大規模な AI ベースのシステムの一部となり、運用の中で使用できる。 モデルは一定時間間隔の設定でスケジュールされたバッチ予測を実行する場合もあれば、リクエストに応じてリアルタイムで実行する場合もある。

モデルのモニタリングとチューニング

モデルを使用している間に状況が変化し、モデルが意図した性能から逸脱する可能性がある（2.3 項及び 7.6 項を参照）。 ドリフトを確実に特定して管理するためには、運用しているモデルを定期的に受け入れ基準に照らして評価する必要がある。

ドリフトの問題に対処するためにモデルを更新する必要があると考えられる場合や、より正確で堅牢なモデルを作成するために新しいデータで再訓練を行う必要があると判断される場合がある。この場合、新しいモデルを作成し、更新された訓練データで訓練を行う。 新しいモデルは、A/B テスト（セクション 9.4 参照）のような形で既存のモデルと比較される場合がある。

図 1 に示す ML ワークフローは、論理的な順序で構成されている。実際には、このワークフローではステップが繰り返し実行される（例えば、モデルを評価する際は訓練ステップに戻ることが多く、場合によってはデータ準備に戻ることもある）。

図 1 に示したステップには、ML モデルとシステム全体の非 ML 部分との統合が含まれていない。 一般的に、ML モデルは単独ではデプロイできず、非 ML 部分と統合する必要がある。例えば、ビジョンアプリケーションでは、ML モデルにデータを入力する前に、データのクリーニングや修正を行うデータパイプラインがある。 モデルがより大きな AI ベースのシステムの一部である場合、デプロイ前にこのシステムに統合する必要がある。 この場合、セクション 7.2 で説明するように、統合テスト、システムテスト、受け入れテストの各レベルを実施する場合がある。

3.3 ML の形態の選択

適切な ML アプローチを選択する際には、以下のガイドラインが適用される。

- 選択した ML アプローチに利用できる、十分な訓練データとテストデータが必要となる。
- 教師あり学習では、適切にラベル付けされたデータが必要となる。
- 出力ラベルがある場合は、教師あり学習の可能性がある。
- 出力が離散的でカテゴリ分類されている場合、それは分類である可能性がある。
- 出力が数値で本質的に連続的であれば、それは回帰である可能性がある。
- 与えられたデータセットに出力が提供されていない場合は、教師なし学習の可能性がある。
- 類似データをグループ化する問題であれば、それはクラスタリングである可能性がある。
- 同時発生するデータ項目の検索に関わる問題であれば、それはアソシエーション分析である可能性がある。

- 強化学習は、環境とのやりとりがあるコンテキストに適している。
- 問題に複数の状態という概念があり、状態ごとに判断を伴う場合には、強化学習が適用できる場合がある。

3.4 ML のアルゴリズム選択に関わる要素

最適な ML アルゴリズム、ML モデルの設定、ML モデルのハイパーパラメータを選択する決定的な方法はない。実際には、次の要素を組み合わせて、このセットを選択する。

- 要求される機能（例えば、機能が離散値の分類なのか、予測なのか）
- 次のような要求される品質特性
    - 精度（例えばあるモデルは精度が高いが速度が遅い場合がある）
    - 利用可能なメモリの制約（例えば組み込みシステムの場合）
    - モデルの訓練（および再訓練）の速度
    - 予測の速さ（例えばリアルタイムシステムの場合）
    - 透明性、解釈可能性、説明可能性についての要求
- モデルの訓練に利用できるデータの種類（例えば一部のモデルは画像データでしか動作しない）
- モデルの訓練とテストに利用できるデータの量（例えば一部のモデルは、限られたデータ量では他のモデルよりもオーバーフィットする傾向がある）
- モデルで使用されると予想される入力データの特徴量の数（例えば速度や精度などの他の要素が特徴量の数に直接影響を受ける可能性がある）
- クラスタリングの対象となるクラスの数（例えば一部のモデルは複数のクラスを持つ問題に適さない場合がある）
- これまでの経験
- 試行錯誤

3.5 オーバーフィッティングとアンダーフィッティング

3.5.1 オーバーフィッティング

オーバーフィッティングは、モデルがデータポイントのセットにフィットしすぎて、適切な一般化ができない場合に発生する。 このようなモデルは、訓練に使用したデータでは非常によく機能するが、新しいデータに対して正確な予測を行うことができない可能性がある。オーバーフィッティングは、ノイズや外れ値と表現されるデータポイントを含む、すべてのデータポイントにモデルを適合させようとする

と発生する可能性がある。また、訓練データのセットに十分なデータが提供されていない場合にも発生する可能性がある。

3.5.2 アンダーフィッティング

アンダーフィッティングは、モデルが訓練データのパターンに正確に適合するほど洗練されていない場合に発生する。 アンダーフィッティングのモデルは、単純すぎる傾向があり、新しいデータと、訓練データに非常に類似したデータの両方に対して正確な予測を行うことができない可能性がある。 アンダーフィッティングの原因の一つとして、入力と出力の間の重要な関係を反映した特徴量が訓練データセットに含まれていない可能性がある。 また、アルゴリズムがデータに正しく合っていない場合にも発生する（例えば非線形データに対して線形モデルを作成する場合）。

3.5.3 ハンズオン演習：オーバーフィッティングとアンダーフィッティングの実演

モデルのオーバーフィッティングとアンダーフィッティングの概念を実演すること。これは、データがほとんど含まれていないデータセット（オーバーフィッティング）と、特徴量の相関性が低いデータセット（アンダーフィッティング）を使用して実演できる。

4. ML - データ - 230 分

キーワード

なし

AI に特化したキーワード

アノテーション、データ拡張、分類モデル、データラベリング、データ準備、ML 訓練データ、教師あり学習、テストデータセット、検証データセット

第 4 章の学習目標

4.1 ML ワークフローの一環としてのデータ準備

AI-4.1.1 K2 データ準備に関連する活動と課題を説明する。

HO-4.1. 1H2 ML モデルの作成に必要なデータの準備を行う。

4.2 ML ワークフローにおける訓練データセット、検証データセット、テストデータセット

AI-4.2.1 K2 ML モデルの開発において、訓練データ、検証データ、テストデータの使用を対比する。

HO-4.2. 1H2 訓練データセットとテストデータセットを識別し、ML モデルを作成する。

4.3 データセットの品質に関する問題

AI-4.3.1 K2 典型的なデータセットの品質問題について説明する。

4.4 データの品質とその ML モデルへの影響

AI-4.4.1 K2 データの品質の低さが、結果として得られる ML モデルにどのような問題を引き起こすかを認識する。

4.5 教師あり学習のためのデータラベリング

AI-4.5.1 K1 教師あり学習のためのデータセットにおけるデータのラベリングに対するさまざまなアプローチを想起する。

AI-4.5.2 K1 データセットのデータが誤ってラベル付けされている理由を想起する。

4.1 ML ワークフローの一環としてのデータ準備

データの準備には、ML ワークフローの平均 43%の労力が費やされ、ML ワークフローの中で最も多くのリソースを必要とする活動と言える。これに対し、モデルの選択と構築にはわずか 17%しか使われていない[R17]。 データ準備は、生データを取り込み、ML モデルの訓練と学習済み ML モデルによる予測の両方に使用できる形式でデータを出力するデータパイプラインの一部を構成する。

データ準備は、次の活動から成り立っていると考えられる。

データ取得

- 識別：訓練や予測に使用するデータの種類を特定する。 例えば自動運転車の場合は、レーダー、ビデオ、LiDAR（Laser Imaging Detection and Ranging）データの必要性の識別が含まれる。
- 収集：データソースを識別し、データ収集の手段を決定する。 例えば、国際通貨基金（IMF）を金融データのソースとして識別し、そのデータを AI ベースのシステムに投入するためのチャンネルを識別することが含まれる。
- ラベリング：セクション 4.5 節参照。

取得したデータは、様々な形式（数値、カテゴリー、画像、表、テキスト、時系列、センサー、地理空間、ビデオ、オーディオなど）にすることができる。

データの前処理

- クリーニング： 誤ったデータ、重複データ、外れ値などが確認された場合、それらを削除または修正する。 また、データ補完を用いて、欠損しているデータ値を推定値または推測値（平均値、中央値、最頻値など）に置き換えることもある。 また、個人情報の削除や匿名化を行うこともある。
- 変換：与えられたデータの形式を変更すること（例：文字列で保持されている住所の構成要素への分解、ランダムな識別子を保持するフィールドの削除、カテゴリーデータの数値データへの変換、画像フォーマットの変換）。 数値データの変換には、同じ範囲を使用するためのスケーリングなどがある。例えば、平均値が 0、標準偏差が 1 になるようにデータを再スケーリングする「標準化」がある。この正規化により、データの範囲が 0 から 1 の間になるようになる。
- データ拡張：データセットに含まれるサンプル数を増やすために使用される。また、敵対的サンプルを訓練データに含めることで、敵対的な攻撃に対するロバスト性を高めることもできる（9.1 節参照）。
- サンプリング：利用可能な全データセットの一部を選択することで、より大きなデータセットのパターンを観察することができる。一般的には、ML モデルを作成するためのコストや時間を削減するために行われる。

すべての前処理には、有効なデータを変更したり、無効なデータを追加したりするリスクがあることに注意する。

特徴量エンジニアリング

- 特徴量選択：特徴量とは、データに反映されている属性/プロパティのことである。特徴量選択には、モデルの訓練や予測に最も貢献する可能性の高い特徴量を選択することが含まれる。 実際には、結果として得られるモデルに影響を与えることが期待されていない（あるいは望まれていない）特徴量を除去することも含まれることが多い。 無関係な情報（ノイズ）を除去することで、特徴量の選択は、全体の訓練時間を短縮し、オーバーフィッティング（セクション 3.5.1 参照）を防ぎ、精度を高め、モデルをより汎化可能にすることができる。
- 特徴量抽出：既存の特徴量から、有益で非冗長な特徴量を抽出することである。結果として得られるデータセットは一般的に小さくなり、同等の精度の ML モデルをより安価かつ迅速に生成するために使用することができる。

これらのデータ準備作業と並行して、データ準備作業全体をサポートするために、探索的データ分析（EDA）が行われるのが一般的である。 これには、データに固有の傾向を発見するためのデータ分析や、データの傾向をプロットすることでデータを視覚的に表現するデータの可視化が含まれる。

上記のデータ準備活動およびサブ活動は論理的な順序で示されているが、プロジェクトによっては順序を変えたり、サブセットのみを使用する場合もある。 データソースの特定など、データ準備のステップの中には、一度だけ実行され、手動で実行できるものもある。他のステップは、運用データパイプラインの一部であり、通常はライブデータで動作する。これらのタスクは自動化する必要がある。

4.1.1 データ準備の課題

データ準備に関する課題には、以下のようなものがある。

- 以下の知識が必要である。
  - アプリケーション・ドメイン
  - データとその特性
  - データの準備に関連する様々なテクニック
- 複数のソースから質の高いデータを得ることの難しさ。
- データ・パイプラインを自動化することの難しさ、およびプロダクション・データ・パイプラインがスケーラブルであり、妥当なパフォーマンス効率（例：データ・アイテムの処理を完了するのに必要な時間）を持つことを保証すること。
- データの準備にかかる費用。
- データ準備中にデータパイプラインに混入した欠陥のチェックを十分に優先していない。
- データバイアスの混入（セクション 2.4 参照）。

4.1.2 ハンズオン演習：ML のためのデータ準備

与えられた生データに対して、セクション 4.1 で説明した該当するデータ準備手順を実行し、教師あり学習を用いて分類モデルを作成するために使用するデータセットを作成する。

この活動は、今後の演習に使用される ML モデルを作成するための最初のステップとなる。

この活動を行うために、受講者に以下のような適切な（そして言語固有の）教材を用意する。

- ライブラリ
- ML フレームワーク
- ツール
- 開発環境

4.2 ML ワークフローにおける訓練、検証、テストデータセット

論理的には、ML モデルを構築するためには、3 つの同等のデータセット（すなわち、1 つの初期データセットからランダムに選択されたデータ）が必要となる。

- モデルの訓練に使用される訓練データセット。
- 検証データセットは、モデルの評価とその後のチューニングに使用される。
- テストデータセット（ホールドアウトデータセットとも呼ばれる）は、調整されたモデルをテストするために使用される。

無制限に適切なデータが利用できる場合、ML ワークフローで訓練、評価、テストに使用されるデータ量は、一般的に以下の要素に依存する。

- モデルの訓練に使用されたアルゴリズム。
- RAM、ディスクスペース、コンピューティングパワー、ネットワーク帯域幅、利用可能な時間などのリソースの可用性。

実際には、適切なデータを十分に取得するのは難しいため、訓練データセットと検証データセットは、1 つの組み合わせたデータセットから得られることが多い。テストデータセットは別にしておき、訓練時には使用しない。これは、開発したモデルがテストデータの影響を受けないようにし、また、テスト結果がモデルの品質を正しく反映するようにするためである。

組み合わせたデータセットを 3 つの個別データセットに分割する最適な比率はないが、ガイドラインとして使用できる典型的な比率は、60:20:20 から 80:10:10（訓練：検証：テスト）である。 これらのデータセットへの分割は、データセットが小さい場合や、結果として得られるデータセットが想定される運用データを代表していないリスクがある場合を除き、ランダムに行われることが多い。

利用可能なデータが限られている場合、利用可能なデータを 3 つのデータセットに分割すると、効果的な訓練を行うために必要なデータが不足する可能性がある。この問題を解決するには、訓練データセッ

トと検証データセットを組み合わせ（テストデータセットは別にする）、このデータセットを複数の分割組み合わせ（例：訓練 80%／検証 20%）にすることができる。その後、データは訓練データセットと検証データセットに無作為に割り当てられる。これらの複数の分割された組み合わせを用いて、訓練、検証、およびチューニングを行い、複数のチューニングされたモデルを作成し、モデル全体の性能をすべての実行の平均値として計算することができる。 複数の分割された組み合わせを作成するには、スプリットテスト、ブートストラップ、K-分割交差検証、一個抜き交差検証など、さまざまな方法がある（詳細は[B02]を参照）。

4.2.1 ハンズオン演習：訓練データとテストデータを特定し、ML モデルを作成する

以前に用意したデータ（4.1.2 項参照）を、訓練データ、検証データ、テストデータに分割する。

これらのデータセットを用いて、教師あり学習を用いた分類モデルを訓練・テストする。

検証データセットとテストデータセットで達成された精度を比較することで、評価/チューニングとテストの違いを説明する。

4.3 データセットの品質問題

データセット内のデータに関する典型的な品質問題には、以下の表に示すものがあるが、これに限定されるものではない。

品質の側面	説明
誤ったデータ	取込んだデータが間違っていた（センサーの故障など）、または間違って入力された（コピー&ペーストのミスなど）。
不完全なデータ	データ値が欠落している可能性がある（例えば、レコードのフィールドが空であったり、特定の時間間隔のデータが省略された場合である）。データが不完全である理由は、セキュリティ上の問題、ハードウェア上の問題、人為的なミスなどさまざまである。
誤ったラベルのデータ	誤ったラベルが付いたデータの原因はいくつか考えられる（4.5.2 節参照）。
不十分なデータ	使用している学習アルゴリズムがパターンを認識するのに必要なデータが不足している（アルゴリズムによって必要最小限のデータ量は異なることに注意）。

前処理されていないデータ	データがきれいで、一貫したフォーマットであり、不要な外れ値が含まれていないことを保証するために、データの前処理を行う必要がある（4.1 節参照）。
古いデータ	学習と予測の両方に使用するデータは、可能な限り最新のものでなければならない（例えば、数年前の金融データを使用すると、不正確な結果になる可能性がある）。
アンバランスなデータ	バランスの取れていないデータは、（人種、性別、民族性などによる）不適切なバイアス、センサーの設置不備（例：顔認識カメラを天井の高さに設置するなど）、データセットの入手可能性のばらつき、データ提供者の動機の違いなどによって生じる可能性がある。
不公平なデータ	公平さは主観的な品質特性であるが、多くの場合、特定することができる。例えば、多様性やジェンダーバランスをサポートするために、選択されたデータがマイノリティや不利な立場にあるグループに積極的に偏っている場合がある（このようなデータは公平であると考えられるが、バランスが取れていない可能性があることに注意する）。
重複したデータ	重複したデータは、結果として得られる ML モデルに過度の影響を与える可能性がある。
関連性のないデータ	取り組んでいる問題に関係のないデータは、結果に悪影響を及ぼし、リソースの無駄遣いにつながる可能性がある。
プライバシーに関する問題	いかなるデータを利用する場合でも、関連するデータプライバシー法（EU における個人情報に関する GDPR など）を尊重する必要がある。
セキュリティ問題	訓練データに意図的に挿入された不正なデータや誤解を招くようなデータは、訓練されたモデルを不正確にする可能性がある。

4.4 データ品質とその ML モデルへの影響

ML モデルの品質は、モデルを作成する際のデータセットの品質に大きく依存する。 データの質が低いと、モデルにも予測にも欠陥が生じる。

以下のカテゴリーの不具合は、データ品質の問題に起因するものである。

- 精度の低下：これらの欠陥は、誤ったデータ、不完全なデータ、誤ったラベル、不十分なデータ、古いデータ、無関係なデータ、および前処理されていないデータによって引き起こされる。例えば、住宅価格の予測モデルを構築するためにデータを使用する場合、訓練データにはコンサバトリー付きの戸建て住宅に関するデータがほとんどあるいは全く含まれていなければ、この特定の住宅タイプの予測価格はおそらく不正確なものになる。
- バイアスのあるモデル：データが不完全、不均衡、不公平、多様性に欠ける、重複しているなどの欠陥がある場合に発生する。 例えば、特定の特徴量を持つデータが欠けている場合（例えば、疾病予測のための医療データがすべて特定の性別の被験者から集められている場合）、結果として得られるモデルに悪影響を及ぼす可能性がある（そのモデルが運用上その性別の予測にのみ使用される場合を除く）。
- 汚染されたモデル：これらの欠陥は、データのプライバシーやセキュリティ上の制約に起因するものである。例えば、データのプライバシーに関する問題は、セキュリティ上の脆弱性につながり、攻撃者がモデルから情報をリバースエンジニアリングすることを可能にし、その後、個人情報の漏洩を引き起こす可能性がある。

4.5 教師あり学習のためのデータラベリング

データラベリングとは、ラベル付けされていない（あるいはラベル付けが不十分な）データにラベルを追加して、教師あり学習に適したデータにすることである． データラベリングはリソース集約的な作業であり、ML プロジェクトでは平均して 25%の時間を費やしていると報告されている[B11]。

最も単純な形では、データラベリングとは、画像やテキストファイルをクラスごとに様々なフォルダに分類することである。例えば、肯定的な製品レビューのテキストファイルはすべて 1 つのフォルダに入れ、否定的なレビューはすべて別のフォルダに入れる。 また、画像内のオブジェクトの周りに四角形を描いてラベル付けすることも、一般的なラベル付けの手法で、アノテーションと呼ばれている。3D オブジェクトのラベル付けや、不規則なオブジェクトの周囲にバウンディングボックスを描く場合は、より複雑なアノテーションが必要になる。データのラベル付けやアノテーションは、通常、ツールを使用する。

4.5.1 データラベリングへの取り組み

ラベリングにはいくつかの方法がある。

- 内部：ラベリングは、開発者、テスト担当者、またはラベリングのために作られた組織内のチームによって行われる。
- 外部委託：外部の専門組織にラベリングを依頼する。
- クラウドソース：大勢の個人によってラベル付けが行われること。ラベリングの品質管理が難しいため、複数のアノテーターに同じデータのラベリングを依頼し、使用するラベルを決定することがある。

- AI が支援する：AI ベースのツールを使用して、データを認識してアノテーションを付けたり、類似したデータをクラスタリングしたりする。 その結果を人間が確認したり、（バウンディングボックスを修正するなどして）補足するという 2 段階のプロセスを経る。
- ハイブリッド：上記のラベリング手法を組み合わせて使用することができる。例えば、クラウドソースによるラベリングは、専門的な AI ベースのクラウドマネジメントツールを利用できる外部組織によって管理されるのが一般的である。

該当するなら、事前にラベル付けされたデータセットを再利用することで、データのラベル付けの必要性を完全に回避できる場合がある。このようなデータセットは、例えば Kaggle[R16]などで公開されている。

4.5.2 データセット内の誤ってラベル付けされたデータ

教師あり学習では、データアノテーターによってデータが正しくラベル付けされていることを前提としている。しかし、実際には、データセットのすべてのアイテムが正しくラベル付けされることは稀である。データが誤ってラベル付けされるのは、以下のような理由からである。

- アノテーターによりランダムなエラーが発生する可能性がある（例：間違ったボタンを押すなど）。
- 系統だったエラーが発生する可能性がある（例：ラベラーが間違った指示を受けたり、不十分なトレーニングを受けた場合など）。
- 悪意のあるデータアノテーターによる意図的なエラーが発生することもある。
- 翻訳エラーにより、ある言語で正しくラベル付けされたデータが、別の言語では誤ってラベル付けされることがある。
- 選択肢に解釈の余地がある場合、データアノテーターの主観的な判断により、アノテーターのデータラベルが一致しない可能性がある。
- 必要なドメインの知識がないと、間違ったラベル付けをしてしまう可能性がある。
- 複雑な分類タスクは、より多くのエラーが発生する可能性がある。
- データのラベリングをサポートするツールには、誤ったラベルにつながる欠陥がある。
- ML ベースのラベリングアプローチは確率的であるため、誤ったラベルを付けてしまうことがある。

5. ML 機能パフォーマンスメトリクス - 120 分

キーワード

なし

AI に特化したキーワード

正解率、AUC、混同行列、F1 スコア、クラスター間メトリクス、クラスター内メトリクス、平均二乗誤差（MSE）、ML ベンチマークスイート、ML 機能パフォーマンスメトリクス、適合率、再現率、ROC(受信者動作特性)曲線、回帰モデル、R2 乗、シルエット係数

第 5 章の学習目標

5.1 混同行列

AI-5.1.1 K3 与えられた混同行列データのセットから、ML 機能パフォーマンスメトリクスを計算する。

5.2 分類、回帰、クラスタリングのための追加の ML 機能パフォーマンスメトリクス

AI-5.2.1 K2 分類、回帰、クラスタリングの ML 機能パフォーマンスメトリクスの背後にある概念を対比し、比較する。

5.3 ML 機能パフォーマンスメトリクスの限界

AI-5.3.1 K2 ML システムの品質を判断するために、ML 機能パフォーマンスメトリクスを使用することの限界を要約する。

5.4 ML 機能パフォーマンスメトリクスの選択

AI-5.4.1 K4 ある ML モデルとシナリオに対して、適切な ML 機能パフォーマンスメトリクスとその値を選択する。

HO-5.4. 1H2 作成した ML モデルを、選択した ML 機能パフォーマンスメトリクスを用いて評価する。

5.5 ML のベンチマークスイート

AI-5.5.1 K2 ML のコンテキストにおけるベンチマークスイートの使用について説明する。

5.1 混同行列

分類問題では、モデルが常に正しく結果を予測することはほとんどない。このような問題では、以下のような混同行列を作ることができる。

		実際	
		陽性	**陰性**
予測	**陽性**	**真陽性 (TP)**	**偽陽性 (FP)**
	陰性	**偽陰性 (FN)**	**真陰性 (TN)**

図 2：混同行列

なお，図 2 の混同行列は，表現方法が異なることもあるが，常に真陽性（TP），真陰性（TN），偽陽性（FP），偽陰性（FN）の 4 つの可能な状況に対する値を示すものである。

混同行列をもとに、以下のメトリクスが定義される。

- 正解率（Accuracy）

  正解率 = (TP + TN) / (TP +TN + FP + FN) * 100

  正解率は、すべての正しい分類の割合を測定する。

- 適合率（Precision）

  適合率＝TP/（TP＋FP）×100

  適合率は，正しく予測された陽性の割合を示す。これは、陽性の予測に対してどれだけ確かかを示す指標である。

- 再現率（Recall）

  再現率= TP / (TP + FN) * 100

  再現率（感度とも呼ばれる）は、実際の陽性に対して正しく予測した割合を示す。これは、どれだけ陽性を見逃していないかを示す尺度である。

- F1 スコア

F1 スコア = 2* (適合率 * 再現率) / (適合率 +再現率)

F1 スコアは、適合率と再現率の調和平均として計算される。 F1 スコアは 0 から 1 の間の値となる。 1 に近いスコアは、誤ったデータが結果にほとんど存在しないことを示している。 F1 スコアが低い場合は、モデルが陽性を検出するのが苦手であることを示している。

5.2 分類、回帰、クラスタリングにおける追加の ML 機能パフォーマンスメトリクス

5.1 節で述べた分類に関連するものに加えて、様々なタイプの ML 問題に対応する数多くのメトリクスがある。 最もよく使われるメトリクスのいくつかを以下に示す。

教師あり分類メトリクス

- ROC（受信者動作特性）曲線とは、二値分類器の識別しきい値を変化させたときの能力をグラフ化したものである。 元々は軍事用レーダーのために開発された手法であることから、このような名前が付けられている。 ROC 曲線は、真陽性率（TPR）（再現率とも呼ばれる）と偽陽性率（FPR = FP / （TN + FP））をプロットしたもので、TPR を y 軸、FPR を x 軸とする。
- AUC (Area Under Curve)は、ROC 曲線の下の面積である。これは分類器の分類能力の度合いを表し、モデルがクラス間をどれだけ区別できるかを示す。AUC が高ければ、モデルの予測はより良いものとなる。

教師あり回帰のメトリクス

教師あり回帰モデルの場合、メトリクスは回帰線が実際のデータポイントにどれだけフィットするかを表す。

- 平均二乗誤差（MSE）は、実際の値と予測値の二乗差の平均値を指す。MSE の値は常に正であり、ゼロに近い値はより良い回帰モデルを示唆している。差を二乗することで、正と負の誤差が互いに相殺されないようにしている。
- R2 乗（決定係数とも呼ばれる）は、回帰モデルが従属変数にどれだけフィットしているかを示すメトリクスである。

教師なしクラスタリングの評価メトリクス

教師なしクラスタリングでは、様々なクラスタ間の距離や、与えられたクラスタ内のデータポイントの近さを表すいくつかのメトリクスがある。

- クラスタ内メトリクス（Intra-cluster metrics）は、クラスタ内のデータポイントの類似性を測定する。
- クラスタ間メトリクス（Inter-cluster metrics）は、異なるクラスタ内のデータポイントの類似性を測定する。

- シルエット係数（シルエットスコアとも呼ばれる）は、クラスタ間距離とクラスタ内距離の平均値に基づいた（-1 から+1 の間の）指標である。 スコアが+1 であれば、クラスタがよく分離されていることを意味し、スコアが 0 であれば、ランダムなクラスタリングを意味し、スコアが-1 であれば、クラスタが誤って割り当てられていることを意味する。

5.3 ML 機能パフォーマンスメトリクスの限界

ML 機能パフォーマンスメトリクスは、モデルの機能性を測定することに限定されている。例えば、正解率、適合率、再現率、MSE、AUC、シルエット係数などである。 これらは、ISO 25010 [S06]で定義されているもの（例：性能効率性）や、第 2 章で述べられているもの（例：説明可能性、柔軟性、自律性）のような、他の非機能的な品質特性を測定するものではない。 本シラバスでは、これらの機能的メトリクスを指す言葉として「パフォーマンスメトリクス」が広く使われていることから、「ML 機能パフォーマンスメトリクス」という言葉を用いている。また、"ML 機能"を加えることで、これらのメトリクスが機械学習に特化したものであり、性能効率性のメトリクスとは関係がないことを強調している。

ML 機能パフォーマンスメトリクスは、他のいくつかの要因によって制約を受ける。

- 教師あり学習では、ラベル付けされたデータに基づいて ML 機能パフォーマンスメトリクスが計算されるため、結果として得られる指標の精度は正しいラベル付けに依存する（4.5 節参照）.
- 測定に使われたデータは代表的なものではない可能性がある（例えば、バイアスが入っている可能性がある）。生成された ML 機能パフォーマンスメトリクスはこのデータに依存する（2.4 節参照）。
- システムが複数のコンポーネントで構成されていても、ML 機能パフォーマンスメトリクスは ML モデルにのみ適用される。 例えば、データパイプラインは、ML 機能パフォーマンスメトリクスでは考慮されない。
- ML 機能パフォーマンスメトリクスの多くは、ツールのサポートがあって初めて測定できるものである。

5.4 ML 機能パフォーマンスメトリクスの選択

通常、混同行列から生成されたすべての ML 機能パフォーマンスメトリクスに対して最高のスコアを達成する ML モデルを構築することはできない。 その代わりに、モデルの使用目的に応じて、最も適切な ML 機能パフォーマンスメトリクスを受け入れ基準として選択する（例えば、偽陽性を最小限に抑えるためには、高い値の適合率が必要であり、偽陰性を最小限に抑えるためには、再現率を高くする必要がある）。 5.1 節および 5.2 節で説明した ML 機能パフォーマンスメトリクスを選択する際には、以下の基準を用いることができる。

- 正解率。このメトリクスは、データセットが対称的である場合（例：偽陽性と偽陰性の数やコストがほぼ同じである場合）に適用できる可能性が高い。このメトリクスは、あるクラスのデータ

が他のクラスよりもずっと多い場合にはあまり適していないが、その場合には F1 スコアを考慮すべきである。

- 適合率。これは、偽陽性のコストが高く、陽性の結果に対する信頼性が高くなければならない場合に適したメトリクスとなる。 スパムフィルター（電子メールをスパムとして分類することが陽性とみなされる）は、実際にはスパムではない多くの電子メールをスパムフォルダーに入れることがほとんどのユーザーに受け入れられないため、高い適合率が必要とされる例である。 分類器が、陽性のケースが非常に大きな割合を占める状況を扱う場合、適合率だけを使用することは良い選択ではない可能性が高い。
- 再現率。陽性を見逃さないことが重要な場合、高い再現率が重要となる。 例えば、がんの検出において、真陽性を見逃して陰性（がんが検出されなかった）としてしまうことは許されない可能性が高い。
- F1 スコア - F1 スコアは、予想されるクラスのバランスが悪く、適合率と再現率が同程度の重要性を持つ場合に最も有効である。

上記のメトリクスに加えて、いくつかのメトリクスがセクション 5.2 で説明されている。 これらは、例えば、一定の ML 問題に適用できる可能性がある。

- ROC 曲線の AUC は、教師あり分類問題に使用することができる。
- MSE と R2 乗は、教師あり回帰問題に使用することができる。
- クラスター間メトリクス、クラスター内メトリクス、およびシルエット係数は、教師なしクラスタリング問題に使用することができる。

5.4.1 ハンズオン演習。作成した ML モデルを評価する

前の演習で訓練された分類モデルを使用して、正解率、適合率、再現率、F1 スコアの値を計算して表示する。 可能であれば、開発フレームワークが提供するライブラリ関数を使用して計算を行う。

5.5 ML 用ベンチマークスイート

新しいデータセット、アルゴリズム、モデル、ハードウェアなどの新しい AI 技術は定期的に発表されており、それぞれの新しい技術の相対的な有効性を判断するのは難しい場合もある。

これらの異なる技術を客観的に比較するために、業界標準の ML ベンチマークスイートが用意されている。 これらは幅広い応用分野をカバーしており、AI や ML のパフォーマンスに関するハードウェアプラットフォーム、ソフトウェアフレームワーク、クラウドプラットフォームを評価するツールを提供している。

ML ベンチマークスイートは、トレーニング時間（例えば、フレームワークが、定義された訓練データセットを用いて、75%の精度などの指定された目標品質メトリクスに ML モデルを訓練するのに要する時間）や推論時間（例えば、訓練された ML モデルが推論を実行するのに要する時間）など、様々な測定量を提供する。

ML のベンチマークスイートは、以下のようないくつかの異なる組織によって提供されている。

- MLCommons［R18］。2020 年に設立された非営利団体で、以前の名称は「ML Perf」といい、ソフトウェアフレームワーク、AI 専用プロセッサ、ML クラウドプラットフォームのベンチマークを提供している。
- DAWNBench [R19]。スタンフォード大学の ML ベンチマークスイート。
- MLMark [R20]。Embedded Microprocessor Benchmark Consortium が提供する、組み込み推論のパフォーマンスと正解率を測定するために設計された ML ベンチマークスイート。

6. ML～ニューラルネットワークとテスト～65 分

キーワード

なし

AI に特化したキーワード

活性値、ディープニューラルネットワーク（DNN）、ML の訓練データ、多層パーセプトロン、ニューラルネットワーク、ニューロンカバレッジ、パーセプトロン、符号変化カバレッジ、符号-符号カバレッジ、教師あり学習、閾値カバレッジ、訓練データ、値変化カバレッジ

第 6 章の学習目標

6.1 ニューラルネットワーク

AI-6.1.1 K2 DNN を含むニューラルネットワークの構造と機能を説明する。

HO-6.1. 1H1 パーセプトロン の実装を体験する。

6.2 ニューラルネットワークのカバレッジ指標

AI-6.2.1 K2 ニューラルネットワークの様々なカバレッジ指標について説明する。

6.1 ニューラルネットワーク

人工ニューラルネットワークは、元々人間の脳の働きを模倣することを目的としており、脳はニューロンが多数結合したものと考えられている。 単層パーセプトロンは、人工ニューラルネットワークの最初の実装例の 1 つであり、1 つの層（すなわち 1 つのニューロン）で構成されるニューラルネットワークである。 単層パーセプトロンは、入力が特定のクラスに属するか否かを判断する分類器の教師あり学習に用いることができる。

現在のニューラルネットワークの多くは、複数の層で構成されていることからディープニューラルネットワークと呼ばれており、多層パーセプトロンと考えることができる（図 3 参照）。

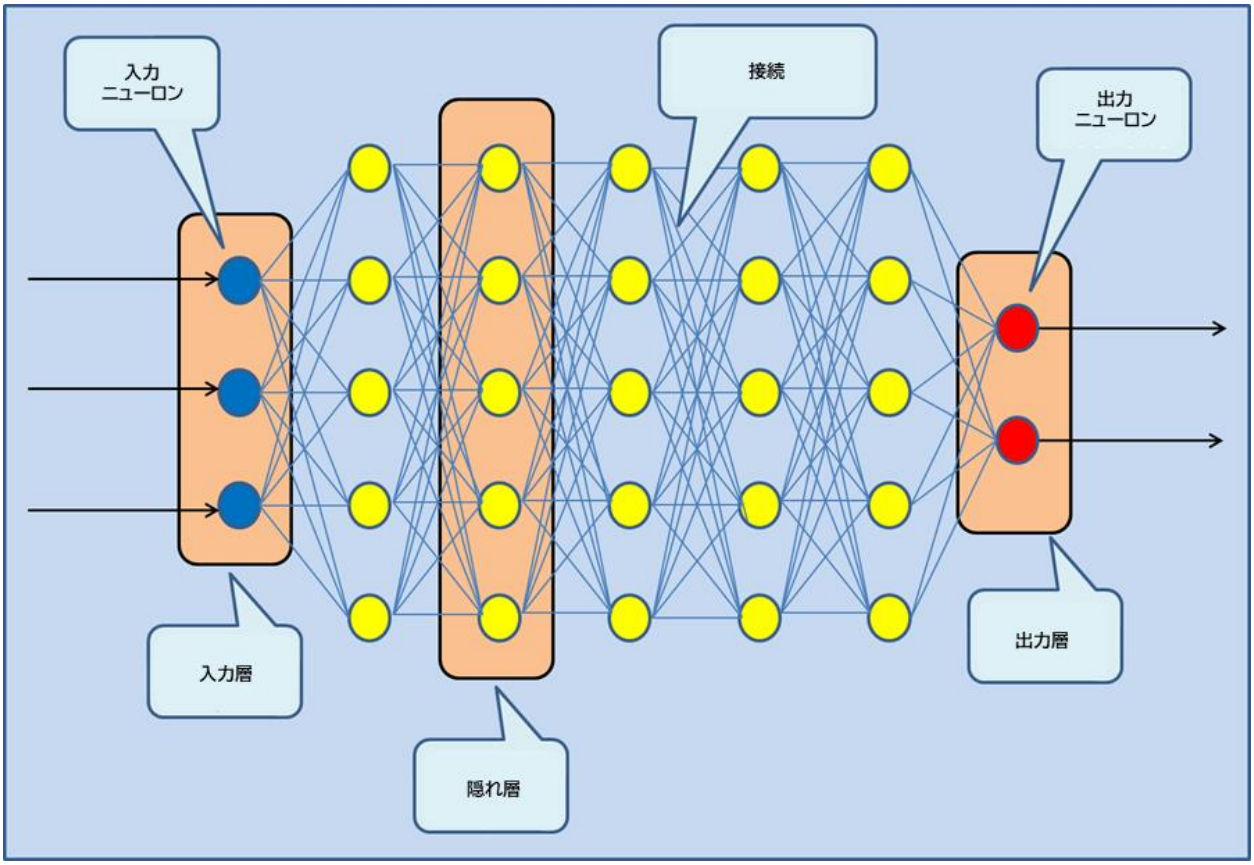

図 3 ディープニューラルネットワークの構造

ディープニューラルネットワークは、3 種類の層で構成されている。 入力層は、カメラからの画素値などの入力を受け取る。 出力層は、結果を外界に提供する。例えば、入力画像が猫である可能性を示す値などが出力される。 入力層と出力層の間には、ノードと呼ばれる人工ニューロンで構成された隠れ層が

ある。 ある層のニューロンは、次の層のニューロンに接続されており、各層のニューロンの数は異なる場合がある。 ニューロンは計算を行い、ネットワーク上で入力ニューロンから出力ニューロンへと情報を伝達する。

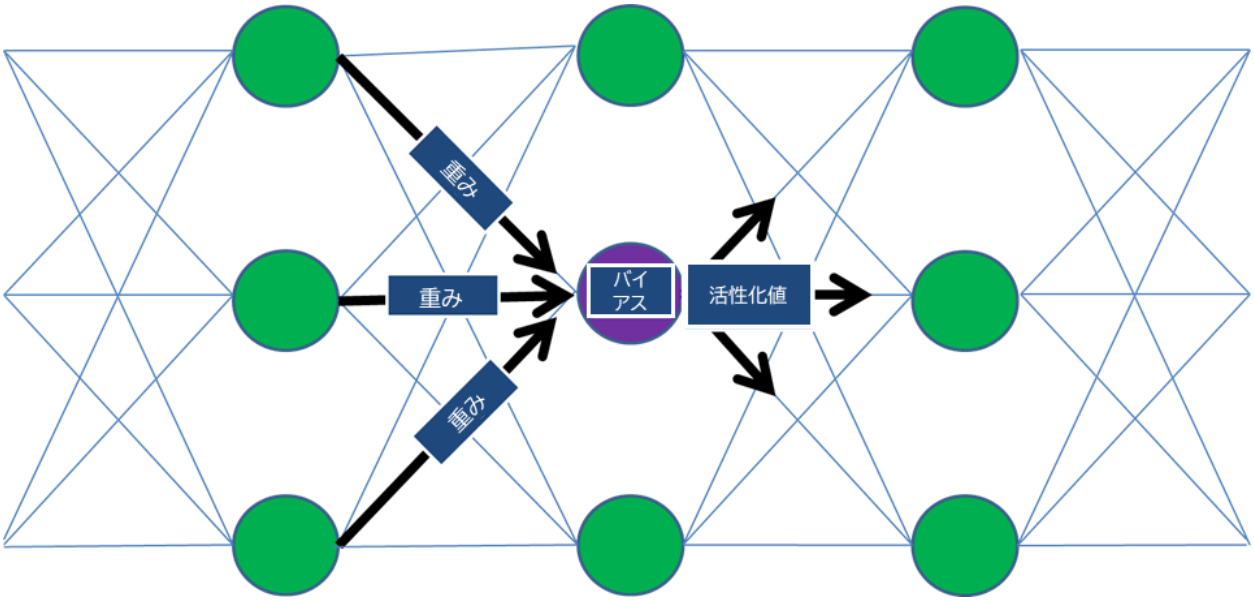

図 4 各ニューロンが行う計算の様子

図 4 に示すように、（入力層のニューロンを除く）各ニューロンが行う計算により、活性化値と呼ばれる値が生成される。この値は、前の層のすべてのニューロンからの活性化値、ニューロン間の接続に割り当てられた重み（これらの重みはネットワークの学習に応じて変化する）、および各ニューロンの個々のバイアスを入力として受け取る式（活性化関数）を実行することによって計算される。ただし，このバイアスはあらかじめ設定された一定の値であり，2.4 節で述べたバイアスとは関係ない。 異なる活性化関数を実行すると、異なる活性化値が計算される。これらの値は通常、ゼロを中心とし、-1（ニューロンが「関心がない」ことを意味する）から+1（ニューロンが「非常に関心がある」ことを意味する）の範囲にある。

ニューラルネットワークの訓練では，各ニューロンにバイアス値を設定し，訓練データをネットワークに与え，各ニューロンに活性化関数を実行させて，最終的に出力を生成する。 生成された出力は、既知の正しい結果と比較される（この教師あり学習の例では、ラベル付きデータが使用される）。 実際の出力と既知の正しい結果との差は、ネットワークを通じてフィードバックされ、この差を最小化するためにニューロン間の接続の重みの値が変更される。 より多くの学習データがネットワークに入力されると、ネットワークの学習に合わせて重みが徐々に調整される。最終的には、生成された出力が十分なものであると判断され、学習が終了する。

6.1.1 ハンズオン演習。シンプルなパーセプトロンの実装

パーセプトロンに AND 関数などの簡単な関数を学習させる演習を行う。

この演習では、パーセプトロンが、誤差がゼロになるまでいくつかのエポックにわたって重みを変更することで学習する方法を取り上げる。この活動には、様々なメカニズム（例：スプレッドシート、シミュレーション）を使用することができる。

6.2 ニューラルネットワークのカバレッジ測定量

ホワイトボックステストのカバレッジ基準（ステートメント、ブランチ、改良条件/決定カバレッジ（MC/DC）[I01]など）を達成することは、従来の命令型ソースコードを使用している場合、一部の安全関連規格[S07]に準拠するために必須であり、その他の重要なアプリケーションについても多くの現場のテスト担当者によって推奨されている。 カバレッジの監視と改善は、新しいテストケースの設計をサポートし、テスト対象に対する信頼性を高めることにつながる。

ニューラルネットワークが実行されるたびに同じコードが実行される傾向があるため、ニューラルネットワークのカバレッジを測定するためにこのような測定量を使用してもほとんど意味がない。 その代わりに、ニューラルネットワーク自体の構造、より具体的にはニューロンのカバレッジに基づいたカバレッジ指標が提案されている。 これらの測定量の多くは、ニューロンの活性化値に基づいている。

ニューラルネットワークに対するカバレッジは、新しい研究分野である。 学術論文が発表されたのは 2017 年以降であり、そのため、提案された測定量が有効であることを示す客観的な証拠（重複する研究結果など）はほとんどない。 ただし、ステートメントカバレッジとディシジョンカバレッジは 50 年以上前から使用されているにもかかわらず、また医療機器や航空電子工学システムなどの安全関連アプリケーションのソフトウェアのカバレッジを測定することが義務付けられているにもかかわらず、それらの相対的な有効性を示す客観的な証拠もほとんどないことに留意する必要がある。

以下のようなニューラルネットワークのカバレッジ基準が提案され、研究者によって様々なアプリケーションに適用されている。

- ニューロンカバレッジ：ニューラルネットワークの各ニューロンがゼロより大きい活性化値を達成することが、フルニューロンカバレッジの条件となる[B12]。 これは実際には非常に簡単に達成でき、様々なディープニューラルネットワークにおいて、非常に少ないテストケースでほぼ 100%のカバレッジを達成したという研究結果がある。 このカバレッジ測定量は、それが達成されない場合のアラーム信号とするのが最も有用であるかもしれない。
- しきい値カバレッジ ：フルしきい値カバレッジでは、ニューラルネットワークの各ニューロンが、指定されたしきい値よりも大きい活性化値を達成する必要がある。DeepXplore フレームワークを開発した研究者は、ニューロンカバレッジは、状況に応じて変化するしきい値を超えた活性化値に基づいて測定すべきだと提案した。彼らは 0.75 のしきい値で研究を行い、このホワイトボックスアプローチを用いて、何千もの誤ったコーナーケースの動作を効率的に見つけたと報告している。 このタイプのカバレッジは、しきい値をゼロに設定したニューロンカバレッジと

容易に区別できるように、ここで名前を変更した。 ある研究者達は、しきい値がゼロのニューロンカバレッジに「ニューロンカバレッジ」という用語を使っている。

- 符号変化カバレッジ：完全な符号変化カバレッジを達成するためには、テストケースは各ニューロンに正と負の両方の活性化値を達成させる必要がある[B13]。
- 値変化カバレッジ：完全な値変化カバレッジを達成するためには、テストケースは各ニューロンに 2 つの活性化値を達成させ、その 2 つの値の差がある選択された値を超える必要がある[B13]。
- 符号-符号カバレッジ: このカバレッジでは、隣接する層のニューロンのペアと、その活性化値が取る符号を考慮する。 ニューロンのペアがカバーされていると考えるためには、テストケースは、第 1 層のニューロンの符号を変更すると、第 2 層のニューロンの符号が変更され、第 2 層の他のすべてのニューロンの符号は変更されないことを示す必要がある[B13]。これは、命令型ソースコードの MC/DC カバレッジと同様の概念である。

研究者たちは，層に基づくさらなるカバレッジ測定量について報告しており（符号-符号カバレッジよりも単純ではあるが），ニューロンの隣接するセットにおける意味のある変化を識別するために最近傍アルゴリズムを使用する成功したアプローチが TensorFuzz ツールに実装されている[B14].

7. AI ベースのシステムのテスト概要 - 115 分

キーワード

入力データテスト、ML モデルテスト

AI に特化したキーワード

AI コンポーネント、自動化バイアス、ビッグデータ、コンセプトドリフト、データパイプライン、ML 機能パフォーマンスメトリクス、訓練データ

第 7 章の学習目標

7.1 AI ベースのシステムの仕様

AI-7.1.1 K2 AI ベースのシステムの仕様が、テストにおいてどのような課題を生み出すかを説明する。

7.2 AI ベースのシステムのテストレベル

AI-7.2.1 K2 AI ベースのシステムが各テストレベルでどのようにテストされるのかを説明する。

7.3 AI ベースのシステムをテストするためのテストデータ

AI-7.3.1 K1 AI ベースのシステムのテストを困難にする可能性のあるテストデータに関する要因を想起する。

7.4 AI ベースのシステムにおける自動化バイアスのテスト

AI-7.4.1 K2 自動化バイアスを説明し、それがテストにどのような影響を与えるかを説明する。

7.5 ML モデルのドキュメント化

AI-7.5.1 K2 AI コンポーネントのドキュメント化について説明し、ドキュメント化が AI ベースのシステムのテストをどのようにサポートするかを理解する。

7.6 コンセプトドリフトのテスト

AI-7.6.1 K2 コンセプトドリフトに対処するために、学習したモデルを頻繁にテストする必要性を説明する。

7.7 ML システムのテスト手法の選択

AI-7.7.1 K4 与えられたシナリオについて、ML システムを開発する際に従うべきテストアプローチを決定する。

7.1 AI ベースのシステムの仕様

システム要件や設計仕様は、AI ベースのシステムでも従来のシステムでも同じように重要である。 これらの仕様は、テスト担当者が実際のシステムの動作が指定された要件に合致しているかどうかを確認するためのベースとなる。 しかし、仕様が不完全でテスト可能性を欠いている場合、テスト・オラクル問題が発生する（8.7 節参照）。

AI ベースのシステムの仕様策定が特に難しいとされる理由はいくつか存在する。

- 多くの AI ベースのシステムを開発するプロジェクトでは、要件はハイレベルなビジネス目標と必要な予測のみで特定される。 その理由は、AI ベースのシステム開発が探索的に行われるからである。 多くの場合、AI ベースのシステムを開発するプロジェクトはデータセットから始まり、そのデータからどのような予測が得られるのかを判断することが目的となる。これは、従来のプロジェクトで最初から必要なロジックを指定するのとは対照的である。
- AI ベースのシステムの精度は、独立したテストによるテスト結果が出るまでわからないことが多い。 探索的な開発手法とともに、望ましい受け入れ基準が決定されたときにはすでに実装が進んでいるため、不十分な仕様になることが多い。
- 多くの AI ベースのシステムは確率的な性質を持っているため、予測の正確さなど、期待される品質要件の一部に許容値を指定する必要がある。
- システムの目標が、特定の機能を提供するのではなく、人間の行動を再現することを求めている場合、システムが代替することを目的とした人間の活動と同等かそれ以上であることを前提とした、不十分な動作要求になってしまうことが度々起こる。 特に、置き換えようとしている人間の能力が大きく異なる場合には、テストオラクルの定義が難しくなる。
- 自然言語認識、コンピュータビジョン、人間との物理的なやりとりなど、ユーザーインターフェースの実装に AI が使用される場合、システムには高い柔軟性が求められる。 しかし、このような柔軟性は、そのようなインタラクションが発生する可能性のあるすべての異なる方法を特定し、ドキュメント化するという課題を生み出す可能性がある。
- 適応性、柔軟性、進化、自律性など、AI ベースのシステムに特有の品質特性を考慮し、要求仕様の一部として定義する必要がある（第 2 章参照）。 これらの特性は新規性が高いため、定義やテストが難しい場合がある。

7.2 AI ベースのシステムのテストレベル

AI ベースのシステムは、通常、AI コンポーネントと非 AI コンポーネントの両方で構成される。 非 AI コンポーネントは、従来のアプローチでテストすることができるが[I01]、AI コンポーネントや AI コンポーネントを含むシステムは、以下に示すように、いくつかの点で異なるテストを行う必要がある。 AI コンポーネントのテストを含むすべてのテストレベルでは、データエンジニア／サイエンティストとドメインエキスパートがテストを密接にサポートすることが重要となる。

従来のソフトウェアのテストレベルとの大きな違いは、AI ベースのシステムで使用される入力データやモデルのテストを明示的に扱うために、新たに 2 つの特化したテストレベルが追加されていることとなる[B15]。 このセクションのほとんどは、すべての AI ベースのシステムに適用できるが、その一部は ML に特化している。

7.2.1 入力データテスト

入力データテストの目的は、システムが学習や予測に使用するデータが最高の品質であることを確認することであり、（セクション 4.3 参照） 以下のような内容となる。

- レビュー
- 統計的手法（例：データの偏りの検証）
- 訓練データの EDA
- データパイプラインの静的・動的テスト

データパイプラインは通常、データの準備を行う複数のコンポーネントで構成されており（4.1 節参照）、これらのコンポーネントのテストには、コンポーネントテストと統合テストの両方が含まれる。トレーニング用のデータパイプラインは、運用予測をサポートするためのデータパイプラインとは全く異なる場合がある。トレーニング用のデータパイプラインは、運用時に使用される完全にエンジニアリングされた自動化されたバージョンとは異なり、プロトタイプと考えられる。 このため、これら 2 つのバージョンのデータパイプラインのテストは全く異なるものになる。しかし、この 2 つのバージョンの機能的な同等性をテストすることも考慮する必要がある。

7.2.2 ML モデルテスト

ML モデルテストの目的は、選択されたモデルが、指定されたかもしれない性能基準を満たしていることを確認することであり、。以下が含まれる。

- ML 機能パフォーマンス基準（5.1 および 5.2 節節を参照）
- トレーニングの速度、予測の速度、使用するコンピューティングリソース、適応性、透明性など、モデル単体に適した ML の非機能受容基準。

また、ML モデルテストは、ML フレームワーク、アルゴリズム、モデル、モデル設定、ハイパ ーパラメータの選択が可能な限り最適に近いことを確認することを目的としている。 必要に応じて、ML モデルテストには、ホワイトボックスカバレッジ基準を達成するためのテストも含まれる場合がある（6.2 節参照）。選択されたモデルは、その後、AI および非 AI の他のコンポーネントと統合される。

7.2.3 コンポーネントテスト

コンポーネントテストは、ユーザーインターフェースや通信コンポーネントなど、モデル以外のコンポーネントにも適用できる従来のテストレベルである。

7.2.4 コンポーネント統合テスト

コンポーネント統合テストは、システムコンポーネント（AI と非 AI の両方）が正しくやりとりすることを確認するために実施される従来のテストレベルである。 このテストでは、データ・パイプラインからの入力が、モデルによって期待通りに受信されること、また、モデルによって生成された予測が、関連するシステムコンポーネント（例：ユーザーインターフェース）へ変換され、それらによって正しく使用されることをテストする。 AI がサービスとして提供されている場合（セクション 1.7 参照）、コンポーネント統合テストの一環として、提供されたサービスの API テストを行うのが一般的である。

7.2.5 システムテスト

システムテストとは、従来のテストレベルのことで、統合されたコンポーネント（AI と非 AI の両方）の完全なシステムが、機能的および非機能的な観点から、運用環境を忠実に反映したテスト環境で期待通りに動作することを確認するために実施される。 このテストは、システムによっては、想定される運用環境での実地試験や、シミュレータ内でのテスト（テストシナリオが危険であったり、運用環境での再現が困難な場合など）形式をとることがある。

システムテストでは、最初に行った ML モデルのテスト結果が、システムに組み込まれたときに悪影響を及ぼさないことを確認するために、ML 機能パフォーマンス基準を再テストする。 このテストは、AI コンポーネントが意図的に変更されている場合（例：DNN を圧縮してサイズを小さくする）には特に重要となる。

システムテストは、システムの非機能要件の多くがテストされるテストレベルでもある。 例えば、ロバスト性をテストするために敵対的なテストを行ったり、システムの説明可能性をテストしたりすることができる。 必要に応じて、ハードウェアコンポーネント（センサーなど）とのインターフェースをシステムテストの一環としてテストすることもある。

7.2.6 受け入れテスト

受け入れテストは、従来のテストレベルであり、完成したシステムが顧客に受け入れられるかどうかを判断するために使用される。 AI ベースのシステムでは、受け入れ基準の定義が難しい場合がある（セクション 8.8 参照）。 AI がサービスとして提供される場合（セクション 1.7 参照）、意図したシステムに対するサービスの適合性や、例えば ML 機能パフォーマンス基準が十分に達成されているかどうかを判断するために、受け入れ試験が必要となる場合がある。

7.3 AI ベースのシステムをテストするためのテストデータ

状況やテスト対象システム（SUT)によっては、テストデータの取得が困難な場合がある。 AI ベースのシステムのテストデータを扱う上で、以下のようないくつかの潜在的な課題がある。

- ビッグデータ（大容量・高速・多種のデータ）は、作成や管理が難しい場合がある。 例えば、大量の画像や音声を高速で消費するシステムでは、代表的なテストデータを作成することが困難な場合がある。

- 入力データは、特に現実世界の出来事を表現する場合、時間の経過とともに変化する必要がある。 例えば、顔認識システムをテストするために記録された写真は、実生活における数年間の人の加齢を表現するために、「エイジング」が必要になることがある。
- 個人情報やその他の機密データは、サニタイズ、暗号化、または編集のための特別な技術が必要な場合がある。また、使用にあたっては法的な承認が必要な場合もある。
- テスト担当者がデータ収集やデータの前処理にデータサイエンティストと同じ実装を使用すると、これらのステップでの不具合が隠蔽されてしまう可能性がある。

7.4 AI ベースのシステムにおける自動化バイアスのテスト

AI ベースのシステムには人間の意思決定を支援するものがある。しかし、人間がこれらのシステムを信頼しすぎてしまう傾向が時折見られる。 これは、自動化バイアス、コンプレイセンシーバイアスと呼ばれ、2 つの形態が存在する。

- 自動化/コンプレインセンシーバイアスの 1 つ目の形態は、人間がシステムから提供された推奨事項を受け入れ、他のソース（自分自身を含む）からの入力を考慮しない場合である。 例えば、人間がデータをフォームに入力する手順は、機械学習を使ってフォームを事前に入力し、人間がこのデータを検証することで改善されるかもしれない。 このような自動化によるバイアスは、通常、意思決定の質を 5%低下させることが示されているが、システムの状況によってはこれよりもはるかに大きくなる可能性がある[B16]。 同様に、携帯電話のメッセージなどで入力されたテキストの自動修正は、しばしば欠陥があるため，その意味を変えてしまう可能性がある。ユーザーはこのことに気づかず、ミスを上書きしないことが多い。
- 自動化/コンプレイセンシーバイアスの 2 つ目の形態は、人間がシステムを十分に監視していないために、システムの故障を見逃してしまうことである。 例えば、半自動運転車は自動運転化が進んでいるが、事故が差し迫っている場合には人間が引き継ぐ必要がある。 一般的に、人間の乗員は、車両を制御するシステムの能力を徐々に信頼しすぎて、注意力が低下していく。そうなると、必要なときに適切な対応ができなくなってしまうことがある。

どちらのシナリオにおいても、テスト担当者は、人間の意思決定がどのように損なわれるかを理解し、システムの推奨事項の品質と、代表的なユーザーが提供する、対応する人間による入力の品質の両方をテストする必要がある。

7.5 AI コンポーネントのドキュメント化

AI コンポーネントのドキュメントの代表的な内容は以下となる。

- 一般的事項：識別子、説明、開発者の詳細、ハードウェア要件、ライセンスの詳細、バージョン、日付、連絡先
- 設計：前提条件、技術的判断

- 使用方法：一次および二次ユースケース、典型的なユーザー、自己学習へのアプローチ、既知のバイアス、倫理的問題、安全性の問題、透明性、意思決定の閾値、プラットフォーム、コンセプトドリフト。
- データセット：特徴量、収集、入手可能性、前処理要件、使用、コンテンツ、ラベル付け、サイズ、プライバシー、セキュリティ、バイアス/公平さ、制限/制約。
- テスト：テストデータセット（説明性と入手可能性）、テストの独立性、テスト結果、ロバスト性、説明可能性、コンセプトドリフト、ポータビリティに関するテスト手法。
- トレーニングと ML 機能パフォーマンス：ML アルゴリズム、重み、検証データセット、ML 機能パフォーマンスメトリクス指標の選択、ML 機能パフォーマンスメトリクスの閾値、実際の ML 機能パフォーマンスメトリクス

明確なドキュメントは、AI を使ったシステムの実装を透明化することで、テストの改善に役立つ。 テストを行う上で重要なドキュメントの分野は以下となる。

- システムの目的、機能的および非機能的な要求事項の仕様。 これらのドキュメントは、通常、テストベースの一部を形成する。
- 異なる AI および非 AI コンポーネントがどのようにやりとりするかを概説したアーキテクチャと設計情報。これは、統合テストの目的の特定をサポートし、システム構造のホワイトボックステストのベースを提供することができる。
- 動作環境の仕様。これは、システムの自律性、柔軟性、適応性をテストする際に必要となる。
- 入力データのソース（関連するメタデータを含む）。このことは以下の点をテストする際には、明確に理解しておく必要がある。
    - 信頼できない入力の機能的正しさ
    - 明示的または暗黙的なデータバイアス
    - 自己学習型システムにおけるデータ入力の不備による誤学習を含めた柔軟性
- システムがその運用環境の変化に適応するために期待される方法。これは、適応性をテストする際のテストベースとして必要である。
- 想定されるシステムユーザーの詳細。これは、代表的なテストを確実に行うために必要となる。

7.6 コンセプトドリフトテスト

運用環境が時間とともに変化しても、学習したモデルがそれに対応して変化することはない。 この現象はコンセプトドリフトと呼ばれ、一般的にモデルのアウトプットの精度や有用性が次第に低下していく。 例えば、マーケティングキャンペーンの影響で、一定期間に渡って潜在的な顧客の行動が変化することがある。 このような変化は、季節的なものであったり、システムの外部にある文化的、道徳的、社

会的な変化による急激な変化であったりする。 このような急激な変化の例としては、COVID-19 パンデミックの影響と、売上予測や株式市場に使用されるモデルの精度への影響が挙げられる。

コンセプトドリフトが発生しやすいシステムは、合意された ML 機能パフォーマンス基準に照らして定期的にテストを行い、コンセプトドリフトの発生を早期に検知して問題を軽減できるようにしなければならない。 典型的な緩和策としては、現行システムの処分や再トレーニングがある。再トレーニングの場合には、最新の訓練データを用いて、確認テスト、リグレッションテスト、場合によっては A/B テスト（セクション 9.4 参照）を実施し、更新された B システムが元の A システムを上回る必要がある。

7.7 ML システムのテスト手法の選択

AI ベースのシステムは、通常、AI コンポーネントと非 AI コンポーネントの両方を含んでいる。 テスト手法は、このようなシステムのリスク分析に基づいており、従来のテストに加えて、AI コンポーネントや AI ベースのシステムに特有の要因に対応するためのより専門的なテストも含まれる。

以下のリストは、ML システムに特有のいくつかの典型的なリスクとそれに対応する軽減策を示している。なお、このリストは限られた例を示しているに過ぎず、ML システムに特有のリスクは、テストによる緩和が必要なものが多くあることに留意すること。

リスク	内容と可能な緩和策
データの品質が期待値よりも低い可能性がある。	本リスクは、いくつかの点で問題になる可能性があり、それぞれ異なる方法で防止することができる（4.4 項参照）。一般的な軽減策としては、レビュー、EDA、動的テストなどが挙げられる。
運用データのパイプラインに不具合のある可能性がある。	本リスクは、個々のパイプラインコンポーネントの動的テストや、パイプライン全体の統合テストによって、部分的に軽減することができる。
モデル開発に使用した ML ワークフローが最適ではない可能性がある。 (3.2 節参照)	本リスクは、次のような原因が考えられる。 ● フォローすべき ML のワークフローに関する事前合意の欠如 ● ワークフローの選択ミス ● ワークフローに従わないデータエンジニア 専門家とのレビューは、間違ったワークフローを選択する可能性を軽減し、より実践的な管理や監査は、ワークフローへの合意と実行の問題を解決する可能性がある。

ML フレームワーク、アルゴリズム、モデル、モデル設定、ハイパーパラメータの選択が最適ではない可能性がある。	本リスクは、意思決定者の専門知識の不足や、ML ワークフローの評価・調整ステップ（またはテストステップ）の実装が不十分であることが原因と考えられる。 専門家とのレビューは、誤った判断をする可能性を軽減し、より良い管理によって、ワークフローの評価とチューニング（およびテスト）のステップが確実に実行されるようになる。
期待する ML 機能パフォーマンス基準は、ML コンポーネントが単独ではその基準を満たしているにもかかわらず、運用上は満たしていないことがある。	本リスクは、モデルのトレーニングとテストに使用されたデータセットが、実際に使用されるデータを代表していないことが原因であると考えられる。 選択されたデータセットを専門家（またはユーザー）がレビューすることで、選択されたデータが代表的なものではないという可能性を軽減することができる。
期待する ML 機能パフォーマンス基準は満たしているが、ユーザーは提供された結果に満足し無い可能性がある	本リスクは、誤ったパフォーマンス基準を選択したことが原因である可能性がある（例えば、高い適合率が必要なのに高い再現性を選択した場合など）。 専門家とのレビューにより、間違った ML 機能パフォーマンスメトリクスを選択する可能性が軽減される可能性や、経験ベースのテストで不適切な基準が特定される可能性もある。　また、コンセプトドリフトによるリスクも考えられるが、その場合は運用システムのテストをより頻繁に行うことで、リスクを軽減することができる。
期待する ML 機能パフォーマンス基準は満たしているが、ユーザーが提供されたサービスに不満を持っている可能性がある。	本リスクは、システムの非機能要求に焦点が当てられていないことが原因である可能性がある。 なお、AI ベースのシステムの品質特性の範囲は、ISO/IEC 25010 に記載されているもの以外にも広がっている（第 2 章参照）。 リスクベースのアプローチで品質特性の優先順位を決め、関連する非機能テストを実施することで、このリスクを軽減することができる。 または、システムテストの一環として、経験ベースのテストによって特定される要因の組み合わせが原因となっている場合もある。第 8 章では、これらの特性をテストする方法についてのガイダンスを提供する。

自己学習型システムが、ユーザーの期待するサービスを提供できていない可能性がある。	本リスクには、様々な理由が考えられ、例として以下のようなものが挙げられる。 ● 自己学習のためにシステムが使用するデータが不適切な場合がある。 このような場合、専門家によるレビューで問題のあるデータを特定することができる。 ● 自己学習した新機能が受け入れられないことにより、システムが故障している可能性がある。 これは、以前の機能との性能比較を含む自動リグレッションテストによって軽減することができる。 ● システムは、ユーザーが想定していない方法で学習している可能性があるが、これは経験ベースのテストで検出できる。
システムがどのように判断しているのか分からず、ユーザーが不満を感じることがある	本リスクは、説明可能性、解釈可能性、または透明性の欠如に起因する可能性がある。これらの特性をテストする方法の詳細については、8.6 節を参照すること。
ユーザーは、データが訓練データと類似している場合にはモデルが優れた予測を行うが、そうでない場合には結果が不十分と感じる可能性がある	本リスクは、オーバーフィッティング（3.5.1 節参照）によるものと考えられるが、これは、訓練データセットから完全に独立したデータセットでモデルをテストするか、経験ベースのテストを行うことで検出できる。

8. AI に特化した品質特性のテスト - 150 分

キーワード

テストオラクル

AI に特化したキーワード

アルゴリズムバイアス、自律型システム、自律性、エキスパートシステム、説明可能性、不適切なバイアス、解釈可能性、LIME 法、ML 学習データ、非決定論的システム、確率論的システム、サンプリングバイアス、自己学習型システム、透明性

第 8 章の学習目標

8.1 自己学習型システムのテストへの挑戦

AI-8.1.1 K2 AI ベースのシステムの自己学習によって生じるテストの課題を説明することができる。

8.2 AI を用いた自律的なシステムのテスト

AI-8.2.1 K2 AI ベースの自律的なシステムがどのようにテストされるかを説明できる。

8.3 アルゴリズム、サンプリング、不適切なバイアスのテスト

AI-8.3.1 K2 AI ベースのシステムのバイアスをテストする方法を説明することができる。

8.4 確率論的・非決定論的な AI ベースのシステムのテストへの挑戦

AI-8.4.1 K2 AI ベースのシステムの確率論的、非決定論的な性質によって生じるテストの課題を説明することができる。

8.5 複雑な AI ベースのシステムのテストするための課題

AI-8.5.1 K2 AI ベースのシステムの複雑さによって生じるテストの課題を説明することができる。

8.6 AI ベースのシステムの透明性、解釈可能性、説明可能性のテスト

AI-8.6.1 K2 AI ベースのシステムの透明性、解釈可能性、説明可能性をどのようにテストすることができるかを説明できる。

HO-8.6.1 H2 説明可能性がテスト担当者の立場でどのように利用できるかを示すために、ツールを使用する。

8.7 AI ベースのシステムのためのテストオラクル

AI-8.7.1 K2 AI ベースのシステムのならではの特性から生じるテストオラクルの作成における課題を説明することができる。

8.8 テスト目的と受け入れ基準

AI-8.8.1 K4 与えられた AI ベースのシステムの AI 特有の品質特性に対して、適切なテスト目的及び受け入れ基準を選択する。

8.1 自己学習型システムのテストへの挑戦

自己学習型システムをテストする際には、以下のようないくつかの潜在的な課題がある（自己学習型システムの詳細については第 2 章を参照）。

- 予期せぬ変更。システムが動作すべき元の要件と制約は一般的に知られているが、システム自体が行った変更については、ほとんど、あるいは全く情報がないことがある。 通常、元の要件や設計（および、規定された制約）に対してテストすることは可能だが、システムが革新的な実装を考案したり、ソリューションを工夫したりした場合（その実装を見ることはできない）、この新しい実装に適したテストを設計することは難しいかもしれない。また、システムが自ら（およびその出力）を変更すると、以前に合格したテストの結果が変わることがある。 これはテスト設計上の課題である。システムの動作が変わっても依然ふさわしいテストを設計することで、リグレッションテストに生じうる問題を防ぐことができるかもしれない。しかし、新しいシステムの動作を観察した上で、新たなテストを設計しなければならない場合もある。
- 複雑な受け入れ基準。システムが自己学習したときに、システムによる改善の期待値を定義することが必要な場合がある。 例えば、システムが自分で変更した場合、そのシステムの全体的な機能パフォーマンスが向上することを想定することができる。 また、単純な「改善」以外のものを指定すると、すぐに複雑になってしまう。 例えば、単純に何かしら改善すればよいではなく、最低限の改善点が指定される場合や、要求される改善が環境要因と関連している場合がある（例：環境要因 F が Y 以上変化した場合、機能 X が最低 10%改善されることが必要）。 これらの問題は、より複雑な受け入れ基準に対する仕様とテスト、および現在のシステムベースラインの機能パフォーマンスの継続的な記録を維持することによって対処することができる。
- テスト時間の不足。さまざまなシナリオを想定した場合に、システムがどのくらいの速さで学習し、適応することができるかを知る必要がある場合がある。 このような受け入れ基準を規定したり、取得したりするのは難しいかもしれない。 システムが迅速に適応する場合、変更のたびに手作業で新しいテストを実行するには時間が足りないかもしれない。そのため、システムの変更時に自動的に実行されるテストを記述する必要があるかもしれない。 これらの課題は、適切な受け入れ基準（セクション 8.8 を参照）の指定と、自動化された継続的なテストによって解決することができる。
- リソース要件。システム要件には、自己学習や適応を行う際にシステムが使用することが許されるリソースの受け入れ基準が含まれる場合がある。これには、例えば、改善のために使用が許可される処理時間とメモリの量が含まる。さらに、このリソースの使用が、機能や精度の測定可能な改善に結びつきそうかどうかについても考慮する必要がある。この課題は、受け入れ基準の規定に影響する。
- 運用環境の仕様が不十分であること。自己学習型システムは、受け取る環境入力が想定範囲外であったり、学習データに反映されていなかったりすると、変化することがある。 これらの入力は、データポイズニング（9.1.2 項参照）の形で攻撃されることもある。 運用環境や環境の変化をすべて予測することは困難であり、そのため代表的なテストケースや環境要件をすべて特定す

ることはできない。 理想的には、システムが対応することが期待される運用環境の可能な変化の全範囲が、受け入れ基準として定義されるべきである。

- 複雑なテスト環境。リスクの高い運用環境の潜在的な変化をすべて模擬できるようにテスト環境を管理することは課題であり、テストツール（フォールトインジェクションツールなど）を使用する場合もある。 運用環境の性質によっては、入力やセンサーを操作してテストしたり、システムをテストできるさまざまな物理的環境にアクセスしてテストしたりすることもある。
- 望ましくない動作の修正。自己学習型のシステムは、入力に基づいて動作を修正するが、テスト担当者がこれを防ぐことができない場合がある。これは、例えば、サードパーティのシステムを使用している場合や、本番のシステムをテストしている場合などに起こる。 自己学習型のシステムは、同じテストを繰り返すことで、そのテストに対応して最適化され、システムの長期的な挙動に影響を与える可能性がある。 そのため、テストによって自己学習型システムの挙動が悪く変化してしまうような事態を防ぐことが重要になる。 これは、テストケースの設計とテスト管理の課題である。

8.2 AI ベースの自律的なシステムのテスト

自律型システムは、人間の介入が必要な場合とそうでない場合を判断できなければならない。 そのため、AI ベースのシステムの自律性をテストするには、システムがこの意思決定を行うための条件を整える必要がある。

自律性をテストするには、以下のことが必要な場合がある。

- 特定のシナリオにおいてシステムが制御を放棄すべき時に、システムが人間の介入を要求するかどうかをテストする。 このようなシナリオには、運用環境の変化や、システムが自律性の限界を超えた場合などが含まれる。
- システムが特定の期間後に制御を放棄すべき時に、システムが人間の介入を要求するかどうかをテストする。
- 本来ならば自律的に動作すべきところを、不必要に人間の介入を求めていないかどうかをテストする。

このテストに必要な条件を生成するために、動作環境に適用される境界値分析を使用するとよい。 自律性を決定するパラメータが動作環境にどのように現れているかを定義し、自律性の性質に応じたテストシナリオを作成することは簡単ではない。

8.3 アルゴリズムバイアス、サンプリングバイアス、不適切なバイアスのテスト

ML システムは、様々なバイアスに対して評価され、不適切なバイアスを取り除くための措置が取られるべきである。これには、不適切なバイアスに対抗するためにポジティブなバイアスを意図的に導入することも含まれる。

独立したデータセットを使ってテストすることで、多くの場合、バイアスを検出することができる。 しかし、ML アルゴリズムは一見無関係な特徴の組み合わせを使って不要なバイアスを発生させることがあるため、バイアスの原因となるすべてのデータを特定するのは難しい場合がある。

AI ベースのシステムは、アルゴリズムバイアス、サンプリングバイアス、不適切なバイアスについてテストする必要がある（セクション 2.4 参照）。 これには以下が含まれる。

- アルゴリズムバイアスがあるかどうかを特定するために、モデルの訓練、評価、チューニングの活動中に分析を行う。
- サンプリングバイアスの存在を特定できるように、訓練データのソースとその取得に使用されたプロセスをレビューする。
- ML のワークフローの一環としてデータの前処理をレビューし、サンプリングバイアスの原因となるような影響を受けていないかどうかを特定する。
- システムの入力の変化がシステムの出力にどのように影響するかを多数のインタラクションで測定し、システムが不適切に偏らせる可能性のある人や物のグループに基づいて結果を検証する。これは、8.6 で説明した LIME（Local Interpretable Model-Agostic Explanations）手法に似ており、リリース前のテストの一環として、本番環境で実施されることがある。
- バイアスに関連する可能性のある入力データの属性に関する追加情報を取得し、結果と関連付ける。 これは、人口統計学的なデータに関連する可能性がある。例えば、人口のグループに影響を与える不適切なバイアスをテストする場合に適していることがある。このとき、そのグループのメンバーはバイアスの評価に使われるが、モデルの入力ではない。 これは、バイアスが、入力データには明示的に存在しないが、アルゴリズムによって推測される「隠れた」変数に基づいている可能性があるためである。

8.4 確率論的・非決定論的な AI ベースのシステムのテストへの挑戦

確率論的なシステムの多くは非決定論的でもあるため、以下のテストの課題は、これらの属性を持つ AI ベースのシステムに典型的に適用される。

- 同じ前提条件と入力を用いたテストでも、複数の有効な結果が得られる可能性がある。これは、期待結果の定義をより困難にし、以下の場合で問題を引き起こす可能性がある。
    - テストが確認テストに再利用される場合

- テストがリグレッションテストに再利用される場合
- テストの再現性が重要な場合
- テストが自動化されている場合

- テスト担当者は、単に期待されるテスト結果の正確な値を述べるのではなく、テストに合格したかどうかを合理的にチェックできるように、要求されるシステムの動作についてより深い知識を必要とする。 例えば、テスト担当者は、従来のシステムと比較して、より洗練された期待値を定義する必要があるかもしれない。この期待されるテスト結果には、許容範囲が含まれることがある（例えば、「実際の結果が最適解の 2%以内に収まっているか」など）。
- システムが確率的な性質を持っているために、テストからの単一の決定的な出力が不可能な場合、統計的に有効なテスト結果を生成するために、テスト担当者がテストを数回実行することがしばしば必要となる。

8.5 複雑な AI ベースのシステムのテストへの挑戦

AI ベースのシステムは、人間が実行するには複雑すぎるタスクの実装によく使われる。 これは、テスト担当者が通常のように期待結果を判断することができないため、テストオラクル問題につながる可能性がある（8.7 項参照）。 例えば、大量のデータからパターンを特定するために、AI ベースのシステムがよく使われる。 このようなシステムが使用されるのは、人間がいくら分析しても手作業では見つけられないようなパターンを見つけることができるからである。このようなシステムに求められる動作を十分に理解して、期待結果を作り出すのは難しいことである。

同様の問題は、AI ベースのシステムの内部構造がソフトウェアによって生成され、人間が理解できないほど複雑になっている場合にも生じる。 これは、AI ベースのシステムがブラックボックスとしてしかテストできないという状況を引き起こす。 このようなケースでは、内部構造が見えていても、テストに役立つ追加情報は得られない。

AI ベースのシステムの複雑さは、確率的な結果を提供したり、非決定論的な性質を持つ場合に増大する（8.4 節参照）。

非決定論的なシステムの問題は、AI ベースのシステムが複数の相互作用するコンポーネントで構成され、それぞれが確率的な結果を提供する場合に悪化する。 例えば、顔認識システムでは、画像内の顔を識別するために 1 つのモデルを使用し、どの顔が識別されたかを認識するために 2 つ目のモデルを使用する可能性がある。 AI コンポーネント間の相互作用は複雑で理解するのが難しいため、すべてのリスクを特定し、システムを適切に検証するテストを設計することが困難になる。

8.6 AI ベースのシステムの透明性、解釈可能性、説明可能性のテスト

システムがどのように実装されているかについての情報は、システム開発者から提供される。 これには、訓練データのソース、ラベリングの実施方法、システムコンポーネントの設計方法などが含まれる。 これらの情報が得られない場合、テストの設計が困難になる可能性がある。 例えば、訓練データ

の情報が得られない場合、そのデータの潜在的なギャップを特定し、そのギャップの影響をテストすることが困難になる。 このような状況は、ブラックボックステストとホワイトボックステストと比較すると、同様のメリットとデメリットがある。 透明性のテストは、データやアルゴリズムに記載されている情報と実際の実装を比較し、それらがどの程度一致しているかを判断することで行うことができる。

ML では、特定の入力と特定の出力の間の関連性を説明することが、従来のシステムに比べて困難な場合がある。 これは、出力を生成するモデル自体がコード（アルゴリズム）によって生成されており、人間の問題意識が反映されていないことが主な理由である。 ML のモデルによって説明可能なレベルが異なるため、説明可能性や試験性など、システムに求められる要件に応じて選択する必要がある。

説明可能性を理解する方法の一つとして、テストデータに摂動を加えたときの ML モデルの動的テストがある。 このような方法で説明可能性を定量化し、視覚的に説明する方法が存在する。 これらの手法の中には、モデルに依存しないものもあれば、特定のタイプのモデルに特化し、そのモデルへのアクセスを必要とするものもある。 探索的テストはモデルの入力と出力の関係をよりよく理解するためにも使用できる。

LIME 法はモデルに依存せず、動的に注入された入力摂動と出力の分析を用いて、入力と出力の関係をテスト担当者に提供する。 これは、モデルの説明可能性を提供するための効果的な手法となる。 しかし、これは決定的な理由ではなく、可能性のある出力の理由を提供することに限定されており、すべてのタイプのアルゴリズムに適用できるわけではない。

AI ベースのシステムの解釈可能性は、それが誰に適用されるかに大きく依存する。 ステークホルダーによって、基礎となる技術をどの程度把握する必要があるかという点で、要求が異なる場合がある。

解釈可能性と説明可能性の両方について、理解のレベルを測定しテストすることは、利害関係者の能力レベルが異なり、同意しない可能性があるため、困難な場合がある。 さらに、典型的な利害関係者のプロファイルを特定することは、多くの種類のシステムでは困難な場合がある。このテストを行う場合、一般的にはユーザー調査やアンケートの形で行われる。

8.6.1 ハンズオンエクササイズ：モデルの説明可能性

先に作成したモデルに基づいて、説明性を提供するために適切なツールを使用する。 例えば、画像分類モデルやテキスト分類モデルの場合は、LIME のようなモデルを問わない方法が適している。

受講者は、このツールを使用して、モデルの決定についての説明を作成する必要がある。特に、入力の特徴が出力にどのように影響するかについて説明する。

8.7 AI ベースのシステムのためのテストオラクル

AI ベースのシステムのテストにおける主要な問題は、期待結果の仕様である。 テストオラクルは、テストの期待結果を決定するために使用されるソースである[I01]。期待結果を決定する際の課題は、テストオラクル問題として知られている。

複雑で非決定論的なシステムや確率論的なシステムでは、「グランドトゥルース」（AI ベースのシステムが予測しようとしている現実世界での実際の結果）を知らずに、テストオラクルを確立することは困

難な場合がある。 この「グランドトゥルース」は、テストオラクルとは異なる。テストオラクルは、必ずしも期待値を提供するものではなく、システムが正しく動作しているかどうかを判断するためのメカニズムを提供するものである。

AI ベースのシステムは進化する可能性があり（セクション 2.3 参照）、自己学習型システム（セクション 8.1 参照）のテストでは、自己学習型システムが自分自身を修正するため、テストオラクル問題に悩まされる可能性があり、それによってシステムの機能的な期待値を頻繁に更新する必要がある。

効果的なテストオラクルを得ることが難しいもう一つの原因は、多くの場合、ソフトウェアの動作の正しさが主観的であることである。 仮想アシスタント（Siri や Alexa など）はこの問題の一例であり、ユーザーによって期待することが全く異なることが多く、言葉の選択や話し方の明瞭さによって異なる結果を経験する可能性がある。

状況によっては、期待結果を制限や許容値で定義することが可能な場合がある。 例えば、自律走行車の停車位置を、特定の地点から最大距離以内と定義することができる。エキスパートシステムでは、専門家に相談して期待結果を決定することができる（専門家の意見が間違っている可能性もある）。 このような状況では、以下のいくつかの重要な要素を考慮する必要がある。

- 人間の専門家は、その能力のレベルが異なる。システムが代替することを意図している専門家と同等以上の能力を持つ専門家が関与する必要がある。
- 同じ情報を提示されても、専門家同士では意見が一致しないことがある。
- 人間の専門家は，自分の判断を自動化することを認めない場合がある。そのような場合には、潜在的なアウトプットの評価は二重盲検でなければならない（つまり、専門家もアウトプットの評価者も、どの評価が自動化されたかを知るべきではない）。
- 人間は、回答に注釈をつける傾向がある（例：「よくわからないが...」のような表現）。 このような注意書きが AI ベースのシステムにない場合、回答を比較する際にはこの点を考慮する必要がある。

A/B テスト（セクション 9.4 参照）、バックツーバックテスト（セクション 9.3 参照）、メタモルフィックテスト（セクション 9.5 参照）など、テストオラクル問題を軽減するテスト技法が存在する。

8.8 テスト目的と受け入れ基準

システムのテスト目的と受け入れ基準は、認識された製品リスクに基づく必要がある。 これらのリスクは、多くの場合、要求される品質特性の分析から特定できる。 AI ベースのシステムの品質特性には、ISO/IEC 25010 [S06]で伝統的に考慮されてきたもの（すなわち、機能適合性、性能効率性、互換性、使用性、信頼性、セキュリティ、保守性、および移植性）が含まれるが、さらに以下の側面についても考慮する必要がある。

特性	受け入れ基準

適応性	● 環境の変化に適応したときに、システムが正しく機能し、非機能要件を満たしていることを確認する。 これは、自動リグレッションテストの一形態として実施されることもある。 ● 環境の変化にシステムが適応するまでの時間を確認する。 ● システムが環境の変化に適応する際に使用されるリソースを確認する。
柔軟性	● 初期の仕様にない状況にシステムがどのように対処するかを検討する。これは、変更された運用環境で実行される自動リグレッションテストの形で実施することができる。 ● 新しいコンテキストに対応してシステムを変更するのにかかる時間や使用するリソースを確認する。
進化	● システムが自分の経験からどれだけ学習しているかを確認する。 ● データのプロファイルが変更された場合（例えば、コンセプトドリフト）、システムがどの程度対応できるかを確認する。
自律性	● 完全に自律すると期待される動作範囲の外に追いやられたときに、システムがどのように反応するかを確認する。 ● 完全に自律すべきシステムが、人間の介入を要求するように「説得」されるかどうかを確認する。
透明性、解釈可能性、説明可能性	● アルゴリズムやデータセットへのアクセスのしやすさを確認することで、透明性を確認する。 ● 解釈可能性と説明可能性は、システムのユーザーに質問することで確認する。実際のシステムのユーザーがいない場合は、同じようなバックグラウンドを持つ人に質問する。
不適切なバイアスの不在	● システムがバイアスの影響を受ける可能性がある場合は、専門家のレビュアーを使う、または、独立したバイアスのないテストスイートを使うことでテストができる。 ● 国による調査などの外部データを用いてテスト結果を比較し、推測される変数に不要なバイアスがかかっていないかを確認する（外部妥当性テスト）。
倫理	● EC Assessment List for Trustworthy Artificial Intelligence（信頼可能な人工知能のための EC 評価リスト）[R21]などの適切なチェックリストに照らし合わせてシステムをチェックする。このチェックリストは、信頼可能な人工知能のための倫理ガイドライン（Ethics Guidelines for

	Trustworthy Artificial Intelligence）[R22]で概説されている主要な要件をサポートしている。
確率論的システムと非決定論的システム	● これは正確な受け入れ基準で評価することはできない。正常に動作していても、同じテストでわずかに異なる結果を返すことがある。
副作用	● 潜在的に有害な副作用を特定し、システムがその副作用を起こすようなテストを生成することを試みる。
報酬ハッキング	● テスト対象の知的エージェントに対して、独立したテストにおいて成功の判断に異なる手段を使用する場合、報酬ハッキングを識別することができる。
安全性	● これは、仮想テスト環境（セクション 10.2 参照）などを使って慎重に評価する必要がある。これには、システムに強制的に危害を加えようとする試みも含まれる。

ML システムでは、ML モデルに要求される ML 機能パフォーマンスメトリクスを規定する必要がある（第 5 章参照）。

9. AI ベースのシステムのテストのための方法と技法 - 245 分

キーワード

A/B テスト、敵対的テスト、バックツーバックテスト、エラー推測、経験ベースのテスト、探索的テスト、メタモルフィック関係（MR）、メタモルフィックテスト（MT）、ペアワイズテスト、疑似オラクル、テストオラクル問題、ツアー

AI に特化したキーワード

敵対的攻撃、敵対的サンプル、データポイズニング、ML システム、学習済みモデル

第 9 章の学習目標

9.1 敵対的攻撃とデータポイズニング

AI-9.1.1 K2 ML システムのテストは、敵対的な攻撃やデータポイズニングの防止にどのように役立つか説明する

9.2 ペアワイズテスト

AI-9.2.1 K2 ペアワイズテストが AI ベースのシステムにどのように使われるか説明する

LO-9.2. 1H2 AI ベースのシステムのテストケースを導出し、実行するために、ペアワイズテストを適用する

9.3 バックツーバックテスト

AI-9.3.1 K2 AI ベースのシステムでバックツーバックテストがどのように使用されるかを説明する

9.4 A/B テスト

AI-9.4.1 K2 A/B テストが AI ベースのシステムのテストにどのように適用されるかを説明する

9.5 メタモルフィックテスト

AI-9.5.1 K3 AI ベースのシステムのテストにメタモルフィックテストを適用する

HO-9.5. 1H2 与えられたシナリオに対するテストケースを導出し、実行するために、メタモルフィックテストを適用する

9.6 AI ベースのシステムにおける経験ベースのテスト

AI-9.6.1 K2 経験ベースのテストが AI ベースのシステムのテストにどのように適用できるか説明する

HO-9.6. 1H2 AI ベースのシステムに探索的テストを適用する

9.7 AI ベースのシステムに対するテスト技法の選択

AI-9.7.1 K4 与えられたシナリオについて、AI ベースのシステムをテストする際に適切なテスト技法を選択する

9.1 敵対的攻撃とデータポイズニング

9.1.1 敵対的攻撃

敵対的攻撃とは、攻撃者が有効な入力を微妙に乱して学習済みモデルに入力することで、誤った予測をさせることを指す。 敵対的サンプルとして知られるこのような乱された入力が最初に注目されたのは迷惑メールのフィルタである。可読性を失わずに迷惑メールをわずかに修正することで騙すことができた。 最近では、画像分類器との関連性が高まっている。 人間の目には見えない数個のピクセルを変更するだけで、ニューラルネットワークを騙して、画像の分類を全く異なる結果に、しかも高い信頼度で変更させることができる。

敵対的サンプルは一般的に転用可能である。つまり、ある ML システムを騙すことができる敵対的サンプルは、同じタスクを実行するように訓練された別の ML システムを騙せることが多いということである 。2 番目の ML システムが異なるデータや異なるアーキテクチャで訓練されていても、同じ敵対的サンプルでは失敗しやすいということである.

ホワイトボックス型の敵対的攻撃とは、攻撃者が、モデルの訓練にどのアルゴリズムが使用されたか、また、モデルにどんな設定やパラメータが使用されたかを知っている場合（一定以上の透明性がある場合）を指す。 攻撃者はこの知識を利用して、例えば、入力に小さな摂動を加え、どの摂動がモデルの出力に大きな変化をもたらすかを監視することで、敵対的サンプルを生成する。

ブラックボックス型の敵対的攻撃では、攻撃者がモデルを探索してその機能を判断し、同様の機能を持つ別のモデルを作成する。 その後、攻撃者はホワイトボックスのアプローチを用いて、作成されたモデルに対する敵対的サンプルを特定する。 敵対的サンプルは一般的に転用可能であるため、ある敵対的サンプルは通常、元のモデルでも敵対的サンプルとして作用する。

同様の機能を持つ別のモデルの作成が不可能な場合は、大量の自動テストで異なる敵対的サンプルを発見し、その結果を観察することが可能な場合がある。

敵対的テストとは、単純に敵対的攻撃を行うことで、脆弱性を特定し、将来の故障の発生を防止するための予防策を講じることを目的としている。 識別された敵対的サンプルは学習データに追加され、モデルがそれらを正しく認識できるように訓練される。

9.1.2 データポイズニング

データポイズニング攻撃とは、攻撃者が訓練データを操作して、2 つの結果のうちの 1 つを得るというものである。攻撃者は、将来の侵入を容易にするためにバックドアやニューラルネットワークのトロイの木馬を挿入することもあれば、より多くの場合、破損した訓練データ（例えば、誤ったラベル付けされたデータ）を使用して、訓練されたモデルが誤った予測を行うように誘導することもある。

ポイズニング攻撃は、特定の状況下で ML システムに誤分類をさせることを目的としている場合がある。 また、サービス妨害攻撃のように、無差別に行う場合もある。 よく知られているポイズニング攻撃の例としては、Microsoft Tay チャットボットへの攻撃がある。これは、比較的少数の有害な Twitter の会話がフィードバックされることで、適切ではない会話を提供するようにシステムを訓練してしまった

ものである。 データポイズニング攻撃でよく使われるのは、数百万通のスパムメールをスパムではないと偽って報告し、スパムフィルターのソフトウェアの判断を歪めようとするものである。 データポイズニングで懸念されるのは、一般に広く利用されている AI データセットがポイズニングを受ける可能性である。

ポイズニングされたデータは外れ値として表示されうるため、データポイズニングを防ぐためのテストを EDA で行うことができる。また、学習データの出所を確実にするため、データ取得ポリシーを見直すこともできる。 運用中の ML システムが、ポイズニングされたデータを入力することで攻撃を受ける可能性がある場合、A/B テスト（セクション 9.4 参照）を用いて、更新されたシステムが以前のものと大きな変化がないことを確認することができる。 また、信頼できるテストスイートを用いて更新されたシステムのリグレッションテストを行うことでも、システムがポイズニングを受けたかどうかを判断することができる。

9.2 ペアワイズテスト

AI ベースのシステムでは、特にビッグデータを利用する場合や、自動運転車のように外界と相互作用する場合には、関係するパラメータの数が非常に多くなることがある。 網羅的なテストを行うためには、これらのパラメータのすべての可能な組み合わせを、すべての可能な値でテストする必要がある。 しかし、そうすると実質的に無限のテストが必要になるため、限られた時間の中で実行可能なサブセットを選択するために、テスト技法が用いられる。

多数のパラメータ（それぞれが多数の離散的な値を持つ可能性がある）を組み合わせることが可能な場合、組み合わせテストを適用することで、必要なテストケースの数を大幅に減らすことができ、それでいて理想的にはテストスイートの欠陥検出能力を損なうこともない。 組み合わせテストの手法はいくつかある（[I02]や[S08]を参照）。しかし、実際には、ペアワイズテストが最も広く使用されている。これは、理解しやすくかつ十分なツールサポートがあるからである。さらに、欠陥の多くは少数のパラメータの相互作用によって引き起こされることが研究で示されている[B33].

実際には、ペアワイズテストを使用しても、システムによっては膨大な数のテストスイートが必要となり、必要な数のテストを実行するためには、自動化や仮想テスト環境（セクション 10.2 参照）の使用が必要となることがある。 例えば、自動運転車を想定した場合、システムテストのためのハイレベルなテストシナリオでは、車の運転が想定されるさまざまな環境と、車のさまざまな機能の両方をテストする必要がある。 そのためには、環境の制約条件（道路の種類や路面、天候や交通状況、視界など）の範囲と、さまざまな自動運転機能（アダプティブクルーズコントロール、車線維持支援、車線変更支援など）をパラメータとみなす必要がある。 これらのパラメータに加えて、センサーからの入力の劣化度合いも考慮する必要がある（例えば、ビデオカメラからの入力は、走行距離が延びるにつれて劣化し、汚くなっていく）。

自動運転車のような安全性が重視される AI ベースのシステムで組み合わせテストを使用する際に必要な厳密さのレベルについては、本シラバス執筆時点では研究中のテーマである。ペアワイズテストでは十分ではない可能性があるが、このアプローチが欠陥の発見に有効であることは知られている。

9.2.1 ハンズオン演習：ペアワイズテスト

実装された AI ベースのシステムで、最低 5 つのパラメータと少なくとも 500 の可能な組み合わせがある場合、ペアワイズテストツールを使用して、削減されたペアワイズの組み合わせのセットを特定し、これらの組み合わせについてテストを実行する。 テストしたペアワイズの組み合わせの数と、理論的に可能な組み合わせをすべてテストした場合に必要な数を比較する。

9.3 バックツーバックテスト

AI ベースのシステムをテストする際のテストオラクル問題（セクション 8.7 参照）に対する潜在的な解決策の 1 つは、バックツーバックテストを使用することである。 これは、差分テストとしても知られている。 バックツーバックテストでは、システムの代替可能なバージョンを疑似的なオラクルとして使用し、その出力を SUT が生成するテスト結果と比較する。 疑似的なものとは、既存のシステムであったり、別のチームが開発したものであったり、異なるプラットフォーム、異なるアーキテクチャ、異なるプログラミング言語であったりする。機能的な適合性（非機能要求ではなく）をテストする場合、疑似的に使用されるシステムは、SUT と同じ非機能的な受け入れ基準を達成するという制約がない。 例えば、実行速度はそれほど速くなくてもよく、その場合、構築費用ははるかに少なくて済む。

ML のコンテキストでは、さまざまなフレームワーク、アルゴリズム、モデル設定を用いて、ML の疑似的なオラクルを作成することが可能である。また、状況によっては、AI ではない従来のソフトウェアを使って疑似的なオラクルを作ることも可能である。

疑似オラクルが欠陥の検出に有効であるためには、疑似オラクルと SUT の両方に共通のソフトウェアが存在してはならない。そうでなければ、両方に同じ欠陥がある場合に、2 つのテスト結果が一致してしまう可能性があるためである。 AI ベースのシステムを開発するために、多くの未成熟で再利用可能なオープンソースの AI ソフトウェアが使用されているため、疑似オラクルと SUT の間でコードを再利用すると、疑似オラクルが適切ではないものとなる可能性がある。また、再利用可能な AI ソリューションのドキュメント化が不十分なため、テスト担当者がこの問題が発生していることを認識するのが難しい場合もある。

9.4 A/B テスト

A/B テストとは、2 つのプログラムのバリエーション（A と B）の同じ入力に対する反応を比較し、どちらのバリエーションが優れているかを判断することを目的とした手法である。 これは統計的なテストアプローチであり、通常、プログラムの違いを判断するためには、複数回のテスト実行によるテスト結果の比較が必要となる。

この方法の簡単な例は、2 つのプロモーションオファーを、2 つのセットに分けたマーケティングリストにメールで送信する場合である。 半分のリストには A のオファー、半分のリストには B のオファーが送られ、それぞれのオファーの成功率によって、今後どちらのオファーを使用するかを決定する。 E コマースやウェブベースの企業の多くは、本番環境で A/B テストを行い、異なる消費者を異なる機能に振り向けることで、消費者の好みを把握している。

A/B テストは、テストオラクル問題を解決するための 1 つのアプローチであり、既存のシステムを部分的なオラクルとして使用する。 A/B テストでは、テストケースは生成されず、テストをどのように設計すべきかについての指針は得られないが、A/B テストにおいて行われた操作はテストにおいて使用されることが多い。

A/B テストは、第 5 章で説明したように、ML 機能パフォーマンスメトリクスなどの受け入れ基準が合意されている AI ベースのシステムのアップデートをテストするために使用できる。 A/B テストは、システムが更新されるたびに、更新されたバージョンが以前のバージョンと同等かそれ以上の性能を発揮するかどうかを確認するために使用される。このようなアプローチは、単純な良しあしの分類にも使えるが、より複雑なシステムのテストにも使える。例えば、スマートシティの交通ルーティングシステムの有効性を向上させるためのアップデートも、A/B テストを用いてテストすることができる（例えば、2 つのバリアントのシステムの平均通勤時間を連続した週に比較するなど）。

A/B テストは、自己学習型のシステムのテストにも利用できる。 システムに変更が加えられると、自動テストが実行され、その結果、システムの特性が変更前のものと比較される。 システムが改善されていれば、その変更は受け入れられ、そうでなければ、システムは以前の状態に戻る。

A/B テストとバックツーバックテストの大きな違いは、A/B テストは同じシステムの 2 つのバージョンを比較することが目的であるのに対し、バックツーバックテストは不具合を検出することが目的であることである。

9.5 メタモルフィックテスト（MT）

メタモルフィックテスト（Metamorphic Testing）[B18]は、合格したソーステストケースに基づいてテストケースを生成することを目的とした技術である。 1 つ以上のフォローアップテストケースは、メタモルフィック関係(MR) に基づいてソース テスト ケースを変更 (メタモルフィック変換) して生成される。MR は、テスト対象が備える機能の特性に基づいており、テストケースのテスト入力の変更が、同じテストケースの期待結果にどのように反映されるかを考慮する。

例えば、ある数字群の平均値を求めるプログラムを考える。 数字の集合と期待される平均値からなるソーステストケースを生成し、テストケースを実行して合格することを確認する。 ここで、プログラムの平均関数についてわかっていることに基づいて、フォローアップのテストケースを生成することができる。 最初は、平均化する数値の順番を変更する。 平均関数が正しく機能していれば、期待される結果は変わらないことが予測できる。 このように、期待結果を計算することなく、数字の順番を変えた後のテストケースを生成することができる。 大規模な数字のセットでは、同じ数字を異なる順序で使用した異なる数字のセットを大量に生成し、それらを使用して多くフォローアップテストケースを作成することができる。これらのテストケースはすべて同じソーステストケースに基づいており、同じ期待結果を持つことになる。

MR やフォローアップテストケースにおいて、期待結果がソーステストケースの期待結果とは異なることがよくある。 例えば、先述の平均関数において、入力セットの各要素を 2 倍にした MR を導出できる。この場合の期待結果は、元の期待結果に 2 を乗じただけのものである。 同様に、任意の他の値を乗算する値として使用し、この MR に基づいて無限のフォローアップテストケースを生成することができる。

MT は、ほとんどのテスト対象に使用することができ、機能テストと非機能テストの両方に適用することができる（例えば、設置性テストでは、インストールパラメータを異なる順序で選択するような異なるターゲット構成をカバーしうる）。 これは安価なテストオラクルの不足によって期待結果の生成に課題がある場合に特に有効である。これは、ビッグデータの分析に基づく AI システムや、ML アルゴリズムがどのようにして予測を導くのかテスト担当者が理解していない場合などに当てはまる。 AI の分野では、画像認識、検索エンジン、経路最適化、音声認識などのテストに MT が使われている。

以上のように、MT は合格したソーステストケースに基づいて行うことが可能であるが、ソーステストケースが正しいかどうかを検証できない場合にも有効である。 これは、例えば、一部の AI ベースのシステムのように、人間のテスト担当者が複製を作ってテストオラクルとして使用するには複雑すぎる機能をプログラムが実装している場合などが考えられる。このような場合、MT を使用して 1 つ以上のテストケースを生成し、それを実行して得られる一連の出力において、出力間の関係の妥当性をチェックすることができる。 この形式の MT では、個々のテストが正しいかどうかはわからないが、それらの間の関係が妥当であることを確認することで、プログラムに対する信頼性が向上する。 例えば、大規模なデータに基づいて死亡年齢を予測する AI ベースの保険プログラムでは、例えば、タバコの喫煙本数を増やすと、予測される死亡年齢が下がる（あるいは、少なくとも変わらない）ことが分かっている。

MT は、1998 年に提案された比較的新しいテスト技法である。従来のテスト技法との違いは、フォローアップテストケースの期待結果が、絶対値ではなく、ソーステストケースの期待結果に対する相対値で記述されることである。 理解しやすい概念に基づいており、アプリケーション領域を理解しているテスト担当者であれば、適用経験が少なくても適用することができ、従来の技法と比較してコストも同程度であることが特徴である。 また、欠陥の発見にも効果的で、従来のテストオラクルに基づいた手法で検出できた欠陥の 90%以上を、わずか 3～6 種類の MR で発見できるという研究結果がある[B19]。 十分な厳密度で記述された MR とソーステストケースからフォローアップテストケースを自動的に生成することは可能である。しかし、現在のところ商用ツールはない。ただし、Google はオープンソース化された GraphicsFuzz ツールを用いて、Android のグラフィックスドライバのテストに自動化された MT を適用している（[R23]参照）。

9.5.1 ハンズオン演習：メタモルフィックテスト

この演習では、受講者は以下のことを実際に体験する。

- 与えられた AI ベースのアプリケーションやプログラムについて、いくつかのメタモルフィック関係（MR）を導き出すこと。これらの MR には、ソーステストケースとフォローアップテストケースの期待結果が同じであるものと、異なるものが含まれている必要がある。
- AI ベースのアプリケーションやプログラムのソーステストケースを作成する。これらは合格することが保証されている必要はないが、受講者は、そのような「ゴールドスタンダード」が利用できない場合があるという、MT の限界を認識する必要がある。
- 導出された MR と生成されたソーステストケースを使って、フォローアップテストケースを導出する。
- フォローアップテストケースを実行する。

9.6 AI ベースのシステムの経験に基づくテスト

経験ベースのテストには、エラー推測、探索的テスト、チェックリストベースのテストなどがあるが[I01]、これらはすべて、AI ベースのシステムのテストにも適用できる。

エラー推測は、一般的に、テスト担当者の知識、開発者の典型的なエラー、類似したシステム（または以前のバージョン）の故障に基づいて行われる。 AI ベースのシステムに適用されるエラー推測の例としては、システム的に偏った訓練データを使用したために ML システムが過去にどのように失敗したかについての知識を使用するものがある。

探索的テストでは、テストの設計、生成、実行が反復的に行われ、初期のテストのテスト結果に基づいて、後のテストを導出する。 探索的テストは、AI ベースのシステムでよく見られるような、仕様書の不備やテストのオラクル問題がある場合に特に有効である。結果として、探索的テストは AI ベースのシステムのテストでよく使用され、メタモルフィックテスト（セクション 9.5 参照）のような技術に基づく比較的体系的なテストを補完するために使用されるべきである。

ツアーとは、テスト担当者が探索的テストを行う際に参照する一連の戦略や目標を表すメタファーで、特定の着目点を中心に構成されている[B20]。 AI ベースのシステムの探索的テストのための典型的なツアーにおいて、ML システムにおけるバイアス、アンダーフィッティング、オーバーフィッティングに着目することがある。 例えば、モデルのテストにデータツアーが適用されるかもしれない。このツアーでは、テスト担当者は、トレーニングに使用されたさまざまな種類のデータ、その分布、バリエーション、フォーマットや範囲などを特定し、そのデータタイプを使用してモデルをテストする。

ML システムは訓練データの質に大きく依存しており、既存の EDA の分野は探索的テストのアプローチと密接に関連している。 EDA は、データのパターン、関係性、傾向、外れ値などを調べるものである。EDA は、対話的に仮説を立ててデータを探索するもので、[B21]では、「予想に基づきデータを探索する。データで見たことに基づいて、予想を修正する。 そして、このプロセスを繰り返す。」と説明されている。EDA では、分析者が複雑なデータを理解するための「データとの対話」と、分析結果を簡単に表示するための「データの可視化」の 2 つの領域でツールのサポートが必要となる。 データの可視化を中心とした探索的な手法を用いることで、使用している ML アルゴリズムの検証や、効率的なモデルにつながる変更点の特定、ドメインの専門知識の活用などに役立てることができる[B22]。

Google は、データ、モデル開発、インフラ、モニタリングの分野で、アサーションとして定義された 28 の ML テストを用意しており、Google 内で ML システムのテストチェックリストとして使われている[B23]。 ここでは、Google が公開している「ML テストチェックリスト」を紹介する。

ML データ

1. 特徴量に関する要件をスキーマで把握している。
2. すべての特徴量が有益である。
3. どの特徴量もコストが高すぎない。
4. 特徴量は、メタレベルの要件を遵守している。

5. データパイプラインには適切なプライバシーコントロールが備わっている。
6. 新しい特徴量を素早く追加できる。
7. すべての入力の特徴量のためのコードがテストされている。

モデル開発

1. モデルの仕様がレビューされ、提出されている。
2. オフラインとオンラインのメトリクスが相関している。
3. すべてのハイパーパラメータがチューニング済みである。
4. モデルの陳腐化の影響がわかっている。
5. シンプルなモデルであれば良いというわけでは必ずしもない。
6. 重要なデータスライスにおいて、モデルの品質が十分である。
7. モデルがプロダクトへの組み込みを考慮してテストされている。

ML のインフラ

1. 訓練には再現性がある。
2. モデルの仕様に基づきユニットテストが実施されている。
3. ML パイプラインの統合テストが実施されている。
4. モデルの品質が提供前に検証されている。
5. モデルはデバッグ可能である。
6. モデルは提供前にカナリアリリースされている。
7. 提供されるモデルはロールバックが可能である。

モニタリングテスト

1. 依存関係の変化が通知される。
2. 入力に対してデータ不変性が成立している。
3. 訓練と提供が偏っていない。
4. モデルが陳腐化していない。
5. モデルが数値的に安定している。
6. コンピューティングパフォーマンスが劣化していない。
7. 予測品質が劣化していない。

9.6.1 ハンズオン演習：探索的テストと探索的データ分析(EDA)

受講者は、選択したモデルとデータセットについて、様々なタイプのデータと、様々なパラメータの分布を考慮しながら、データツアーを行う。

受講者はデータに対して EDA を行い、データの欠落やバイアスの可能性を確認する。

9.7 AI ベースのシステムのためのテスト技法の選択

AI ベースのシステムは、通常、AI コンポーネントと非 AI コンポーネントの両方を含む。 非 AI コンポーネントをテストするためのテスト技法の選択は、一般的に従来のテストと同じである。 AI ベースのコンポーネントでは、その選択はより制約されうる。 例えば、テストオラクルの問題が認識されている場合（すなわち、期待結果を生成することが困難である場合）、認識されたリスクに基づいて、次のような技法によりこの問題を軽減することが可能である。

- バックツーバックテスト：これには、テストケースが用意されている、または、生成されていることと、疑似的なオラクルとして機能する同等のシステムが必要である。リグレッションテストの場合、そのシステムは前バージョンのシステムを使用できる。 欠陥を効果的に検出するためには、独立して開発されたシステムが必要になる場合がある。
- A/B テスト：運用上の入力をテストケースとして使用することが多く、通常は統計的な分析を用いて同じシステムの 2 つのバリエーションを比較するために使用される。 A/B テストは、新しいバリアントのデータポイズニングのチェックや、自己学習型システムの自動リグレッションテストなどに使用される。
- メタモルフィックテスト：これは、経験の浅いテスト担当者が、アプリケーションドメインを理解する必要があるものの、コスト効率よく欠陥を発見するために使用することができる。MT は、期待結果が絶対的なものではなく、ソーステストケースに対する相対的なものであるため、決定的な結果を提供するのには適していない。商用ツールのサポートは現在のところないが、多くのテストを手動で生成することができる。

敵対的テストは、通常、敵対的サンプルの取り扱いを誤ると大きな影響を与える可能性がある ML モデルや、システムが攻撃される可能性がある ML モデルに適している。 同様に、データポイズニングのテストも、システムが攻撃される可能性のある ML システムに適していることがある。

AI ベースのシステムが複雑で複数のパラメータを持つ場合は、ペアワイズテストが適切な場合が多い。

経験に基づくテストは、AI ベースのシステムのテストに適していることが多く、特に訓練データや運用データの検討に適している。 EDA は、使用されている ML アルゴリズムの検証、効率の改善点の特定、ドメインの専門知識の活用に役立てられる。Google は、同社の ML テストチェックリストが ML システムに有効なアプローチであることを発見した。

ニューラルネットワークのうちの特定の分野では、ネットワークのカバレッジがミッションクリティカルなシステムに適していることが多く、カバレッジの基準によっては他の基準よりも厳密なカバレッジが求められることもある。

10. AI ベースのシステムのテスト環境 - 30 分

キーワード

仮想テスト環境

AI に特化したキーワード

AI 専用プロセッサ、自律型システム、ビッグデータ、説明可能性、マルチエージェントシステム、自己学習型システム

第 10 章の学習目標

10.1 AI ベースのシステムのテスト環境

AI-10.1.1 K2 AI ベースのシステムのテスト環境と、従来のシステムに必要なテスト環境を区別する主な要因を説明できる。

10.2 AI ベースのシステムをテストするための仮想テスト環境

AI-10.2.1 K2 AI ベースのシステムのテストにおいて、仮想テスト環境が提供する利点を説明できる。

10.1 AI ベースのシステムのためのテスト環境

AI ベースのシステムは様々な運用環境で使用されるため、そのテスト環境も同様に様々なものがある。テスト環境が従来のシステムと異なる原因となる AI ベースのシステムの特性として、以下のようなものがある。

- 自己学習。自己学習型システムおよび一部の自律型システムは、システムが最初にデプロイされる時には完全に定義されていない可能性のある運用環境の変化に適応することが求められる（セクション 2.1 参照）。そのため、これら未定義の環境変化を模倣できるテスト環境を定義することは本質的に困難であり、テスト担当者の想像力とテスト環境に組み込まれたランダム要素の両方が必要となる場合がある。
- 自律性。自律型システムは、人間の介入なしに環境の変化に対応するとともに、人間に自律性を戻すべき状況を認識することが期待されている（2.2 項参照）。 システムによっては、自律性を戻す状況を特定し模倣するために、テスト環境がシステムを極限まで追い込む必要がある。自律型システムの中には、危険な環境下で動作することを目的とするものがあり、その代表的な危険なテスト環境を設定することは困難な場合がある。
- マルチエージェンシー。マルチエージェント AI ベースのシステムが他の AI ベースのシステムと協調して動作することが期待される場合、テスト環境は、SUT が相互作用する AI ベースのシステムの非決定性を模倣できるように、ある程度の非決定性を組み込む必要がある場合がある。
- 説明可能性。AI ベースのシステムは、その性質上、システムがどのように決定を行ったのかを判断することが困難な場合がある（2.7 項参照）。デプロイ前にこれを理解することが重要な場合、テスト環境には、決定がどのように行われるかを説明する手段としてツールを組み込まなければならない場合がある。
- ハードウェア。AI ベースのシステムをホストするために使用されるハードウェアの中には、AI 専用プロセッサのように、AI の目的のために特別に設計されたものがある（1.6 項参照）。 このようなハードウェアをテスト環境に含めることは、関連するテスト計画の一部として検討する必要がある。
- ビッグデータ。AI ベースのシステムがビッグデータ（例：大量、高速、および／または多種多様のデータ）を消費すると予想される場合、テスト環境の一部としてビックデータを設定するには、慎重な計画と実施が必要である（7.3 項参照）。

10.2 AI ベースのシステムをテストするための仮想テスト環境

AI ベースのシステムをテストする際に仮想テスト環境を使用すると、以下のメリットがある。

- 危険なシナリオ。これらは、SUT、人間を含む他の相互作用するシステム、または運用環境（木や建物など）を危険にさらすことなくテストできる。

- 異常なシナリオ。実際の運用ではこのようなシナリオを設定するにはかなりの時間やコストがかかる場合でも、これらはテストできる（例えば、皆既日食や 4 台のバスが同時に同じ道路の交差点に入るなどの珍しいイベント）。 同様に、現実世界では実現が困難なエッジケースも、仮想テスト環境ではより簡単に、繰り返し、再現性を持って作成できる。
- 極端なシナリオ。これらは、現実ではコストがかかるか不可能な場合でもテストできる（例えば原子力災害や深宇宙探査）。
- 時間のかかるシナリオ。これらは、仮想環境の中でタイムスケールを短縮して（例えば 1 秒間に数回）テストできる。これに対して、実時間でテストするには数時間から数日かかる場合がある。 さらに、複数の仮想テスト環境を並行して実行できるという利点もある。これは通常、クラウド上で実行され、実際のシステムハードウェアでは不可能な、多くのシナリオを同時に実行できる。
- 観測容易性と制御容易性。仮想テスト環境では、格段に優れたテスト環境の制御容易性を提供する。例えば、異常な一連の金融取引の条件を確実に再現できる。また、デジタルで提供される環境のすべての部分を継続的に監視および記録できるため、格段に優れた観測容易性が得られる。
- 可用性：仮想テスト環境でハードウェアをシミュレーションすることにより、開発されていない、または高価であるといった理由で利用できないハードウェアコンポーネントがあっても、シミュレートされたものによってシステムをテストできる。

仮想テスト環境は、特定のシステムに特化して構築される場合もあれば、汎用的である場合もあり、また特定のアプリケーション・ドメインをサポートするために開発される場合もある。 AI ベースのシステムのテストをサポートするために、商用およびオープンソースの仮想テスト環境が用意されている。その例を以下に挙げる。

- Morse：The Modular Open Robots Simulation Engine は、Blender のゲームエンジン[R24]をベースにした、単一または複数のロボットの汎用的なモバイルロボットシミュレーションのためのシミュレータである。
- AI Habitat：Facebook AI が開発したシミュレーションプラットフォームで、写真のようなリアルな 3D 環境で、具現化されたエージェント（仮想ロボットなど）を訓練するために設計されている[R25]。
- DRIVE Constellation：NVIDIA 社が提供する自動運転車向けのオープンでスケーラブルなプラットフォームである。クラウドベースのプラットフォームをベースにしており、数十億マイルの自律走行テストを行うことが可能である[R26]。
- MATLAB と Simulink：これらは、学習データを準備し、ML モデルを生成し、合成データを使用するモデルを含む AI ベースのシステムの実行をシミュレートする機能を提供する[R27]。

11. テストに AI を使う - 195 分

キーワード

ビジュアルテスト

AI に特化したキーワード

ベイジアン手法、分類、クラスタリングアルゴリズム、欠陥予測、グラフィカルユーザーインターフェース（GUI）

第 11 章の学習目標

11.1 テストのための AI 技術

AI-11.1.1 K2 ソフトウェアテストで使用される AI 技術を分類する。

HO-11.1.1 H2 AI が使われにくいテスト活動について、例を挙げて説明する。

11.2 報告された欠陥の分析に AI を活用する

AI-11.2.1 K2 新たに発見された欠陥の分析を支援するために、AI がどのように役立つかを説明する。

11.3 テストケース作成への AI の活用

AI-11.3.1 K2 AI がどのようにテストケース生成を支援するかを説明する。

11.4 リグレッションテストスイートの最適化における AI の活用

AI-11.4.1 K2 リグレッションテストスイートの最適化を AI がどのように支援するかを説明する。

11.5 AI を使った欠陥予測

AI-11.5.1 K2 AI がどのように欠陥予測を支援するか説明する。

HO-11.5.1 H2 簡単な AI ベースの欠陥予測システムを導入する。

11.6 ユーザーインターフェースのテストに AI を活用する

AI-11.6.1 K2 ユーザーインターフェースのテストでの AI の使用について説明する。

11.1 テストのための AI 技術

1.4 節では、いくつかの AI 技術を紹介しているが、これらはすべて、ソフトウェアテストの特定の側面をサポートするために使用することができる。 Harman[B24]によると、ソフトウェアエンジニアリングのコミュニティは、AI 技術の 3 つの大まかな領域を使用している。

- ファジー論理と確率的手法：これは、確率的な問題を扱うために AI 技術を使用するものである。 例えば、AI はベイズ法を用いてシステムの故障を分析・予測することができる。これらは、コンポーネントや機能が故障する可能性を推定したり、人間とシステム間における相互作用の潜在的なランダム性を反映したりする。
- 分類、学習、予測：この分野は、プロジェクト計画の一環としてのコスト予測や、欠陥の予測など、様々なユースケースに利用可能である。ML に代表されるように、この分野は、欠陥管理（11.2 節参照）、欠陥予測（11.5 節参照）、ユーザーインタフェースのテスト（11.6 節参照）など、多くのソフトウェアテストのタスクに利用されている。
- 計算機による探索と最適化技術：大規模かつ複雑な可能性を秘めた探索空間を計算によって探索し、最適化問題を解決するために使用される（例： 探索アルゴリズムの使用）。 例えば、テストケースの生成 (11.3 節参照)、与えられたカバレッジ基準を達成する最小数のテストケースの特定、リグレッションテストケースの最適化 (11.4 節参照) などがある。

上記の分類は、AI が実施可能なテストタスクと異なる AI 技術の間にかなりの重複があるため、必然的に大まかなものとなる。 また、これは一つの分類に過ぎず、他にも同様に有効な分類があるかもしれない。

11.1.1 ハンズオン演習：テストにおける AI の活用について

議論の一環として、受講者は、AI として実装するには現在実用的ではないテスト活動やタスクを特定する。 これらは以下のようなものが考えられる。

- テストオラクルの指定
- ステークホルダーとのコミュニケーションにより、曖昧さを解消し、不足情報を回収
- ユーザーエクスペリエンスの向上を提案
- ステークホルダーの仮定に挑み、意地悪な質問を投げかける
- ユーザーニーズの把握

いくつかの限定されたタスクに使用できる特化型 AI と、現時点で実現できていない汎用型 AI（1.2 節参照）を区別する必要がある。

11.2 報告された欠陥の分析に AI を活用

報告された欠陥は通常、分類され、優先順位が付けられ、重複しているものが特定される。 この活動は、欠陥のトリアージや分析と呼ばれることが多く、欠陥の解決に費やす経過時間を最適化することを目的とする。 AI は、以下のような様々な方法でこの活動をサポートするために使用することができる。

- カテゴリー化：NLP[B25]を使用して欠陥報告書のテキストを分析し、影響を受ける機能の領域などのトピックを抽出し、他のメタデータと一緒に k 近傍法やサポートベクターマシンなどのクラスタリングアルゴリズムに使用することができる。 これらのアルゴリズムは、適切な欠陥カテゴリを特定し、類似した欠陥や重複した欠陥をハイライトする。 AI によるカテゴリー化は、自動化された欠陥報告システム（例えば Microsoft Windows や Firefox 向け）や、多くのソフトウェアエンジニアが参加する大規模プロジェクトで特に有効となる。
- クリティカル性：最もクリティカルな欠陥の特徴量を学習した ML モデルは、報告された欠陥の大きな割合を占めるそれらのシステム障害を引き起こす可能性が最も高い欠陥を特定するために使用することができる[B26]。
- 割り当て：ML モデルは、欠陥の内容や以前の開発者の割り当てに基づいて、特定の欠陥を修正するのに最も適した開発者を提案することができる[B26]。

11.3 テストケースの生成への AI 活用

AI によるテストの生成は、テスト資産を迅速に作成し、カバレッジ（コードや要件の網羅性）を最大化するための非常に効果的な手法である。 これらのテストを生成するための基礎となるのは、ソースコード、ユーザーインターフェース、および機械的に読み取り可能なテストモデルである。 また、ツールによっては、インスツルメンテーションやログファイルを使ってシステムの低レベルの動作を観察することでテストを行うものもある[B27]。

しかし、要求される動作を定義したテストモデルがテストの基礎として使用されない限り、この形式のテスト生成は、一般的にテストオラクルの問題に悩まされることになる。なぜなら、AI ベースのツールは、与えられたテストデータのセットに対して期待される結果を知らないからである。 一つの解決策は、適切なシステムが擬似的なオラクルとして使用できる場合、バックツーバックテストを使用することである（9.3 節参照）。 また、「アプリケーションが応答しない」、「システムがクラッシュしない」などの単純な欠陥指標が発生しないことを期待してテストを実行する方法もある。

AI ベースのテスト生成ツールと類似の非 AI ファズテストツールを比較した研究によると、AI ベースのツールは、同等のレベルのカバレッジを達成し、より多くの欠陥を見つけることができる一方で、欠陥を引き起こすために必要な平均的な一連のステップを平均約 15,000 ステップから約 100 ステップに減らすことができる。これにより、デバッグがはるかに容易となる[B27]。

11.4 リグレッションテストスイートの最適化に AI を活用

システムに変更が加えられると、新しいテストが作成、実行され、リグレッションテストスイートの候補となる。 リグレッションテストスイートが大きくなりすぎないように、頻繁に最適化を行い、テストケースの選択、優先順位付け、さらには追加を行い、より効果的で効率的なリグレッションテストスイートを作成する必要がある。

AI ベースのツールは、例えば、過去のテスト結果、関連する不具合、最新の変更点などの情報を分析することで、リグレッションテストスイートの最適化を行うことができる。例えば、頻繁に欠陥を引き起こしている機能や、最近の変更で影響を受けたテスト実行コードなどが該当する。

研究によると、ほとんどの欠陥を検出したまま、リグレッションテストスイートのサイズを 50%削減することが可能であり[B28]、継続的な統合テストにおいて欠陥検出率を大幅に低下させることなく、テスト実行期間を 40%削減することが可能である[B29]。

11.5 AI による欠陥予測

欠陥予測は、欠陥が存在するかどうか、いくつの欠陥が存在するか、あるいは欠陥が見つかるかどうかを予測するために使用できる。この機能は、使用するツールの洗練さに依存する。 結果は通常、テストの優先順位付けに使用される（例：より多くの欠陥が予測されるコンポーネントにはより多くのテストを行う）。

欠陥予測は、通常、ソースコードのメトリクス、プロセスのメトリクス、人や組織のメトリクスに基づいて行われる。 考慮すべき潜在的な要因が非常に多いため、これらの要因と欠陥の可能性との関係を判断することは人間の能力を超えている。そのため、一般的には ML を使用した AI ベースのアプローチを使用する必要がある。 不具合の予測は、似たような状況（例えば、同じコードベースや同じ開発者）での過去の経験に基づいて行うのが最も効果的である。

ML を用いた欠陥予測は、いくつかの異なる状況で成功している（例：[B30]、[B31]）。 最良の予測因子は、コード行数やサイクロマティックな複雑さといったより広く使われているソースコードの指標ではなく、人や組織の指標であることがわかっている.

11.5.1 ハンズオン演習：欠陥予知システムの構築

受講者は、適切なデータセット（例：ソースコード測定値と対応する欠陥データを含む）を使用して、簡単な欠陥予測モデルを構築し、それを使用して類似コードのソースコード測定値を使用して欠陥の可能性を予測する。

モデルにはデータセットから少なくとも 4 つの特徴量を使用し、選択した特徴量に応じて結果がどのように変化するかを強調するために、クラスはいくつかの異なる特徴量を使用して結果を調べるべきである。

11.6 ユーザーインターフェースのテストに AI を活用

11.6.1 AI を使って GUI（グラフィカルユーザーインターフェース）でテストする

GUI を使ったテストは、コンポーネントテストを除くと、マニュアルテストの典型的な手法であり、テスト自動化の取り組みの出発点となることも多い。結果として、テスト対象物に対する人間のインタラクションをエミュレートしたテストが行われる。 このスクリプトによるテスト自動化は、ユーザーインターフェイス要素の実際の座標軸、またはインターフェイスのソフトウェア定義オブジェクト/ウィジェットのいずれかを使用して、キャプチャ/プレイバックのアプローチを適用することで実装できる。 しかし、このアプローチには、インターフェースの変更、コードの変更、プラットフォームの変更に対する感度など、オブジェクトの識別に関するいくつかの欠点がある。

AI は、様々な基準（XPath、label、id、class、X/Y 座標など）を用いて正しいオブジェクトを識別し、歴史的に最も安定した識別基準を選択するために、AI ベースのツールを採用することで、このアプローチの脆さを軽減することができる。 例えば、アプリケーションの特定のエリアにあるボタンの ID は、リリースごとに変わる可能性があるため、AI ベースのツールは、時間の経過とともにこの ID の重要性を低くし、他の基準により高い依存度を設定するかもしれない。このアプローチでは、ユーザーインターフェース内のオブジェクトを、テストにマッチするもの、またはテストにマッチしないものとして分類する。

一方、ビジュアルテストでは、画像認識を利用して実際のユーザーと同じインターフェースで GUI オブジェクトを操作するため、基盤となるコードやインターフェースの定義 にアクセスする必要がない。これにより、完全に非侵入型となり、基盤となる技術に依存しないこととなる。スクリプトは、目に見えるユーザーインターフェースを通じてのみ動作する。 このアプローチにより、テスト担当者は、画面全体のレイアウトに影響されることなく、人間の実ユーザーと同じように、画面上の画像、ボタン、テキストフィールドを直接操作するスクリプトを作成することができる。 テスト自動化における画像認識の使用は、必要なコンピューティングリソースによって制限されることがある。 しかし、高度な画像認識をサポートする AI が安価に入手できるようになったことで、このアプローチを主流にすることが可能となった。

11.6.2 AI による GUI のテスト

ML モデルは、ユーザーインターフェース画面の受け入れ可能性を判断するために使用することができる（例えば、ヒューリスティックや教師あり学習を使用して）。 これらのモデルに基づいたツールは、誤ってレンダリングされた要素を特定したり、あるオブジェクトがアクセスできないかどうか、または検出しにくいかどうかを判断したり、GUI の視覚的外観に関するその他のさまざまな問題を検出することができる。

画像認識はコンピュータビジョンアルゴリズムの一形態だが、他の形態の AI ベースのコンピュータビジョンを使用して、画像（スクリーンショットなど）を比較し、レイアウト、サイズ、位置、色、フォント、またはオブジェクトのその他の可視属性に対する意図しない変更を識別することができる。 これら

の比較の結果は、テストオブジェクトへの変更がユーザーインターフェースに悪影響を与えていないことを確認するためのリグレッションテストに利用できる。

画面の受け入れ可能性をチェックする技術は、比較ツールと組み合わせることで、より洗練された AI ベースのリグレッションテストツールを作成することができる。このツールは、検出されたユーザーインターフェースの変更がユーザーに受け入れられる可能性が高いかどうか、あるいはその変更にフラグを立てて人間がチェックすべきかどうかをアドバイスすることが可能である。 このような AI ベースのツールは、同じアプリケーションのユーザーインターフェースが様々なブラウザ／デバイス／プラットフォーム上で正しく動作することを確認することを目的とした、異なるブラウザ／デバイス／プラットフォーム上での互換性のテストをサポートするためにも使用できる。

12. リファレンス

12.1 規格

[S01] ISO/IEC TR 29119-11:2020, ソフトウェアおよびシステム工学-ソフトウェアテスト-パート 11 AI ベースのシステムのテストに関するガイドライン

[S02] DIN SPEC 92001-1, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Part 1:Quality Meta Model, https://www.din.de/en/wdc-beuth:din21:303650673 (accessed May 2021).

[S03] DIN SPEC 92001-2, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Part 2: Technical and Organizational Requirements, https://www.din.de/en/innovation-and-research/din-spec-en/projects/wdc-proj:din21:298702628 (accessed May 2021).

[S04] ISO 26262 - https://www.iso.org/standard/68383.html (accessed May 2021)

[S05] ISO/PAS 21448：2019 自動車－意図した機能の安全性 https://webdesk.jsa.or.jp/link/webstore/Book/html/jp/cor/isopas21448_2019_cor.pdf

[S06] システム及びソフトウェア製品の品質要求及び評価（SQuaRE）−システム及びソフトウェア品質モデル https://kikakurui.com/x25/X25010-2013-01.html

[S07] 自動車－機能安全－第 6 部：ソフトウェアレベルにおける製品開発 https://webdesk.jsa.or.jp/books/W11M0090/index/?bunsyo_id=ISO+26262-6%3A2018

[S08] ISO/IEC/IEEE 29119-4:2015, ソフトウェアおよびシステム工学-ソフトウェアテスト-第 4 部：テスト技法 s.

12.2 ISTQB®ドキュメント

[I01] ISTQB®認定テスト担当者ファンデーションレベルシラバス、バージョン 2018 V3.1 https://www.istqb.org/downloads/category/2-foundation-level-documents.html（2021 年 5 月アクセス）。

[I02] ISTQB ®認定テスト担当者アドバンストレベルシラバス　テストアナリスト、バージョン 3.1.1.J03　section 3.2.6 https://jstqb.jp/dl/jstqb.jpdlJSTQB-SyllabusALTA_V311.J03.pdf (2022 年 12 月アクセス).

[I03] ISTQB ®Certified Tester AI Testing, Overview of Syllabus, Version 1.0

12.3 書籍・記事

[B01] カドワラドル、キャロル(2014). "Are the robots are about to rise?Google の新しいエンジニアリングディレクターはそう考えている..."The Guardian. Guardian News and Media Limited., https://www.theguardian.com/technology/2014/feb/22/robots-google-ray-kurzweil-terminator-singularity-artificial-intelligence (accessed May 2021).

[B02] スチュアート・ラッセル、ピーター・ノーヴィグ、人工知能。A Modern Approach, 4th Edition, Pearson, 2020.

[B03] M.Davies ら，"Advancing Neuromorphic Computing With Loihi:A Survey of Results and Outlook," Proceedings of the IEEE, vol. 109, no.5, pp.911-934, May 2021, doi: 10.1109/JPROC.2021.3067593.

[B04] Chris Wiltz, Can Apple Use Its Latest AI Chip for More Than Photos?, Electronics & Test, Artificial Intelligence, https://www.designnews.com/electronics-test/can-apple-use-its-latest-ai-chip-more-photos/153617253461497 (accessed May 2021).

[B05] HUAWEI Reveals the Future of Mobile AI at IFA 2017, Huawei Press Release, https://consumer.huawei.com/en/press/news/2017/ifa2017-kirin970/ (accessed May 2021).

[B06] 個人データの処理に関する自然人の保護及び当該データの自由な移動に関する、及び指令 95/46/EC（一般データ保護規則）の廃止に関する欧州議会及び理事会の REGULATION (EU) 2016/679, 2016 年　4 月,https://eur-lex.europa.eu/eli/reg/2016/679/oj（2021 年 5 月アクセス

[B07] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806, SAE, ,https://www.sae.org/standards/content/j3016_201806/ (accessed May 2021).

[B08] 貿易とデジタル経済に関する G20 大臣会合声明:Annex. Available from: https://www.mofa.go.jp/files/000486596.pdf (accessed May 2021).

[B09] Concrete Problems in AI Safety, Dario Amodei (Google Brain), Chris Olah (Google Brain), Jacob Steinhardt (Stanford University), Paul Christiano (UC Berkeley), John Schulman (OpenAI), Dan Man´e (Google Brain), March 2016. https://arxiv.org/pdf/1606.06565（2021 年 5 月アクセス）。

[B10] Explainable AI: the basics, Policy briefing, Issued:2019 年 11 月 DES6051, ISBN: 978-1-78252-433-5, The Royal Society.

[B11] The Ultimate Guide to Data Labeling for Machine Learning, www.cloudfactory.com/data-labeling-guide (accessed May 2021).

[B12] Pei et al, DeepXplore:Automated Whitebox Testing of Deep Learning Systems, Proceedings of ACM Symposium on Operating Systems Principles (SOSP '17), Jan 2017.

[B13] Sun et al, Testing Deep Neural Networks, https://www.researchgate.net/publication/323747173_Testing_Deep_Neural_Networks, accessed Nov 2019 (accessed May 2021).

[B14] A. Odena and I. Goodfellow, TensorFuzz:Debugging Neural Networks with Coverage-Guided Fuzzing, ArXiv e-prints, Jul. 2018, https://arxiv.org/pdf/1807.10875 (accessed May 2021)

[B15] Riccio, V. et al, Testing Machine Learning based Systems:A Systematic Mapping. Empirical Software Engineering, https://link.springer.com/article/10.1007/s10664-020-09881-0 (accessed May 2021)

[B16] Baudel, Thomas et al, Addressing Cognitive Biases in Augmented Business Decision Systems., https://arxiv.org/abs/2009.08127 (accessed May 2021)

[B17] Papernot, N. et al, Transferability in Machine Learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277, 2016. https://arxiv.org/pdf/1605.07277（2021 年 5 月アクセス）。

[B18] Chen et al, Metamorphic Testing:A Review of Challenges and Opportunities, ACM Comput. Surv. 51, 1, Article 4, January 2018. https://www.researchgate.net/publication/322261865_Metamorphic_Testing_A_Review_of_Challenges_and_Opportunities（2021 年 5 月アクセス）。

[B19] Huai Liu, Fei-Ching Kuo, Dave Towey, and Tsong Yueh Chen., How effectively does metamorphic testing alleviate the oracle problem?, IEEE Transactions on Software Engineering 40, 1, 4-22, 2014

[B20] James Whittaker, Exploratory Software Testing:テスト設計を導くためのヒント、コツ、ツアー、テクニック、第 1 版、Addison-Wesley Professional, 2009.

[B21] L. Wilkinson, A. Anand, and R. Grossman. 高次元ビジュアルアナリティクス. 高次元ビジュアル分析：点分布のペアワイズビューによるインタラクティブな探索. また，本稿では，「高次 元ビジュアル分析：点分布のペアワイズビューが導くインタラクティブな探索」をテーマとした. https://www.cs.uic.edu/~wilkinson/Publications/sorting.pdf (accessed May 2021).

[B22] Ryan Hafen and Terence Critchlow, EDA and ML - A Perfect Pair for Large-Scale Data Analysis, IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6651091 (accessed May 2021).

[B23] Breck, Eric, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley., The ML Test Score:A Rubric for ML Production Readiness and Technical Debt Reduction, IEEE International Conference on Big Data (Big Data), 2017, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8258038（2021 年 5 月アクセス）。

[B24] Harman, The Role of Artificial Intelligence in Software Engineering, In First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), pp.1-6. IEEE, June 2012, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6227961 (accessed May 2021).

[B25] Nilambri et al, A Survey on Automated Duplicate Detection in a Bug Repository, International Journal of Engineering Research & Technology (IJERT), 2014, https://www.ijert.org/research/a-survey-on-automated-duplicate-detection-in-a-bug-repository-IJERTV3IS041769 (2021 年 5 月アクセス)

[B26] Kim, D.; Wang, X.; Kim, S.; Zeller, A.; Cheung, S.C.; Park, S. (2011). "Which Crashes Should I Fix First?Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts," in the IEEE Transactions on Software Engineering, volume 37, https://ieeexplore.ieee.org/document/5711013 (accessed May 2021).

[B27] Mao et al, Sapienz: multi-objective automated testing for Android applications, Proceedings of the 25th International Symposium on Software Testing and Analysis, July 2016, http://www0.cs.ucl.ac.uk/staff/K.Mao/archive/p_issta16_sapienz.pdf (accessed May 2021).

[B28] Rai et al, Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base, World Applied Sciences Journal 31 (4): 654-662, 2014, https://www.researchgate.net/publication/336133351_Regression_Test_Case_Optimization_Using_Honey_Bee_Mating_Optimization_Algorithm_with_Fuzzy_Rule_Base（2021 年 5 月アクセス）。

[B29] Dusica Marijan, Arnaud Gotlieb, Marius Liaaen.A learning algorithm for optimizing continuous integration development and testing practice, Journal of Software :Practice and Experience, Nov 2018.

[B30] Tosun et al, AI-Based Software Defect Predictors:Applications and Benefits in a Case Study, Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference (IAAI-10), 2010.

[B31] Kim et al, Predicting Faults from Cached History, 29th International Conference on Software Engineering (ICSE'07), 2007.

[B32] ナガッパン 2008 Nagappan et al, The Influence of Organizational Structure on Software Quality:The Influence of Organizational Structure on Software Quality: An Empirical Case Study, Proceedings of the 30th international conference on Software Engineering (ICSE'08), May 2008.

[B33] Kuhn 他，Software Fault Interactions and Implications for Software Testing，IEEE Transactions on Software Engineering vol.30, no.6, (June 2004) pp.418-421.

12.4 その他のリファレンス

以下の参考文献は、インターネット上で入手可能な情報を示している。これらの文献は出版時に確認されているが、その文献が利用できなくなった場合、ISTQB は責任を負わない。®

[R01] Wikipedia contributors, "AI effect," Wikipedia, https://en.wikipedia.org/wiki/AI_effect (accessed May 2021).

[R02] https://mxnet.apache.org/（2021 年 5 月アクセス）。

[R03] https://docs.microsoft.com/en-us/cognitive-toolkit/（2021 年 5 月アクセス）。

[R04] IBM Watson, https://www.ibm.com/watson/ai-services

[R05] https://www.tensorflow.org/（2021 年 5 月アクセス）。

[R06] https://keras.io/（2021 年 5 月アクセス）。

[R07] https://pytorch.org/（2021 年 5 月アクセス）。

[R08] https://scikit-learn.org/stable/whats_new/v0.23.html（2021 年 5 月アクセス）。

[R09] NVIDIA VOLTA, https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/ (accessed May 2021).

[R10] Cloud TPU, https://cloud.google.com/tpu/ (accessed May 2021).

[R11] Edge TPU, https://cloud.google.com/edge-tpu/ (accessed May 2021).

[R12] Intel® Nervana™ Neural Network processors deliver the scale and efficiency demanded by deep learning model evolution, https://www.intel.ai/nervana-nnp/ (accessed May 2021).

[R13] The Evolution of EyeQ, https://www.mobileye.com/our-technology/evolution-eyeq-chip/ (accessed May 2021).

[R14] ImageNet - http://www.image-net.org/ (accessed May 2021).

[R15] Google's BERT - https://github.com/google-research/bert (accessed May 2021).

[R16] https://www.kaggle.com/datasets（2021 年 5 月アクセス）。

[R17] https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-data-science-survey（2021 年 5 月アクセス）。

[R18] MLCommons - https://mlcommons.org/ (accessed May 2021).

[R19] DAWNBench - https://dawn.cs.stanford.edu/benchmark (accessed May 2021).

[R20] MLMark - https://www.eembc.org/mlmark (accessed May 2021).

[R21] https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment| Shaping Europe's digital future (europa.eu) (accessed August 2021)

[R22] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai（2021 年 8 月アクセス

[R23] Google GraphicsFuzz, https://github.com/google/graphicsfuzz (accessed May 2021).

[R24] http://www.openrobots.org/morse/doc/0.2.1/morse.html（2021 年 5 月アクセス）。

[R25] https://ai.facebook.com/blog/open-sourcing-ai-habitat-a-simulation-platform-for-embodied-ai-research/（2021 年 5 月アクセス）。

[R26] https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/（2021 年 5 月アクセス）。

[R27] https://uk.mathworks.com/discovery/artificial-intelligence.html#ai-with-matlab（2021 年 5 月アクセス）。

1 付録 A - 略語

略称	説明
AI	人工知能
AIaaS	AI as a service
API	アプリケーション・プログラミング・インターフェース
AUC	エリアアンダーカーブ
DL	ディープラーニング
DNN	ディープニューラルネットワーク
EDA	探索的データ解析
EU	欧州連合
FN	偽陰性
FP	偽陽性
GDPR	一般データ保護規則
GPU	グラフィカル・プロセッシング・ユニット
GUI	グラフィカル・ユーザー・インターフェース
LIME	局所的に解釈可能なモデル-拮抗する説明
MC/DC	修正条件判定カバレッジ
ML	機械学習
MR	メタモルフィック関係
MSE	平均二乗誤差
MT	メタモルフィックテスト
NLP	自然言語処理
ROC	受信機動作特性
SUT	テスト中のシステム
SVM	サポートベクターマシン
TN	真のネガティブ
TP	真のポジティブ

XAI	Explainable AI

2 付録 B - AI 固有の用語とその他の用語

用語の名称	定義
精度	分類器の評価に用いられる ML の機能的な性能指標で，予測が正しかった割合を測定する（ISO/IEC TR 29119-11 以降）
活性化関数	ニューラルネットワークのニューロンに関連付けられた数式で、ニューロンへの入力からニューロンの出力を決定する。
アクティベーション値	ニューラルネットワーク内のニューロンの活性化関数の出力
敵対的攻撃	敵対的な例を意図的に使用して ML モデルを失敗させること
AI as a Service (AIaaS)	AI および AI 開発サービスが集中的にホストされるソフトウェア・ライセンスおよびデリバリー・モデル
AI コンポーネント	AI 機能を提供するコンポーネント
AI 効果	これまでラベル付けされていた AI システムが、技術の進歩により AI とは言えなくなった場合の状況（ISO/IEC TR 29119-11）
AI によるシステム	1 つまたは複数の AI コンポーネントを統合したシステム
AI 専用プロセッサ	AI アプリケーションを加速するために設計された専用ハードウェアの一種
アルゴリズムバイアス	ML アルゴリズムに起因するバイアスの一種
アノテーション	分類のためのラベル付きデータを提供するために、画像内のオブジェクトをバウンディングボックスで識別する活動
曲線下面積（AUC）	分類器が 2 つのクラスをどの程度区別できるかを示す指標。
人工知能（AI）	知識や技能を獲得、処理、創造、応用する工学的システムの能力（ISO/IEC TR 29119-11

協会	サンプル間の関係性や依存性を識別する教師なしの学習手法
データ拡張	既存のデータセットに基づいて新しいデータポイントを作成する活動
自動化バイアス	自動化された意思決定システムの推奨事項を、他の情報源よりも人が好むことで生じる一種の偏り。 類義語：コンプラシーズ・バイアス
自律システム	人の手を介さずに持続的に動作することができるシステム
自律性	人が介在しなくても持続的に動作するシステムの能力(ISO/IEC TR 29119-11)
ベイズモデル	モデルの入力と出力の両方の不確実性を、確率を用いて表現する統計モデル
ベイズの手法	統計モデルのパラメータとして、前後の確率分布を考慮する手法
バイアス	特定の対象物、人、またはグループを他と比較して取り扱う際の体系的な差異（ISO/IEC DIS 22989
ビッグデータ	量、種類、速度、変動性などの特徴を持つ膨大なデータを処理するには、専門的な技術や手法が必要となる。
ケースベースドリーミング	過去の類似した問題の解決策に基づいて新しい問題を解決する手法
チャットボット	文字や音声で会話をするためのアプリケーション。
分類	与えられた入力に対して出力クラスを予測する ML 関数の一種である（ISO/IEC TR 29119-11 以降
クラシファイア	分類に使われる ML モデル 類義語：分類モデル
クラスタリング	類似したデータポイントをグループ化する ML 関数の一種

クラスタリング・アルゴリズム	ML のアルゴリズムの一種で、類似したオブジェクトをクラスターに分類するために用いられる。
コンセプトドリフト	ユーザーの期待や行動、運用環境の変化によって、ML モデルの予測精度の認識が時間とともに変化すること。
コンフュージョン・マトリクス	分類アルゴリズムの ML 機能性能を要約する手法
データ取得	ML モデルで解決すべきビジネス上の問題に関連するデータを取得する活動
データラベリング	ML での分類を支援するために、生データのオブジェクトに意味のあるタグを付加する活動
データパイプライン	ML アルゴリズムによる学習や ML モデルによる予測をサポートする入力データを提供するためのデータ準備活動の実施
データポイント	データセットの一部として使用される、1 つの観測値を構成する 1 つまたは複数の測定値のセット。
データポイズニング	学習データや ML モデルへの入力データを意図的かつ悪意を持って操作すること。
データの準備	ML のワークフローにおけるデータ取得、データの前処理、フィーチャーエンジニアリングの活動について
データの前処理	ML のワークフローにおけるデータクリーニング、データ変換、データ拡張　、データサンプリングなどの活動について
データビジュアライゼーション	データの関係性やトレンド、パターンをグラフで表現する手法。
データセット	ML のトレーニング、評価、テスト、予測に使用されるデータの集合体
判断基準	予測関数の結果を、その値を上回るか下回るかの二値に変換する値 Synonym: 識別のしきい値

デシジョンツリー	ノードが意思決定を表し、ブランチが可能な結果を表すツリー状の ML モデル
演繹的分類法	入力データに推論と論理を適用して分類する装置
ディープラーニング（DL）	多層構造のニューラルネットワークによる ML
ディープニューラルネットワーク	数層のニューロンで構成されるニューラルネットワーク Synonym: マルチレイヤーパーセプトロン
欠陥予測	テスト対象物の中で欠陥が発生する領域や、発生した欠陥の量を予測する技術のこと。
決定論的システム	与えられた入力と開始状態のセットから、同じセットの出力と最終状態を生成するシステム
エッジコンピューティング	分散型アーキテクチャの中で、情報が使用される場所の近くで情報処理が行われる部分。
エポック	全学習データセットに対する ML 学習の繰り返し
エボリューション	より低い、より単純な、または悪い状態から、より高い、より複雑な、またはより良い状態へと継続的に変化するプロセス
エキスパートシステム	人間の専門知識に基づいて開発された知識ベースから推論を行い、特定のドメインやアプリケーション領域の問題を解決する AI ベースのシステム
説明可能性	AI ベースのシステムがどのようにして与えられた結果を導き出したのかを理解するレベル（ISO/IEC TR 29119-11
説明可能な AI（XAI）	AI システムの出力に影響を与える要因を理解することに関する研究分野
探索的データ解析（EDA）	フィーチャーエンジニアリングをサポートするために使用されるデータを、インタラクティブに、仮説に基づいて、視覚的に探索しする

F1 スコア	リコールとプレシジョンのバランスがとれた、分類器を評価するための ML 機能のパフォーマンス指標
偽陰性（FN）	ML モデルの予測で、モデルが誤ってネガティブクラスを予測してしまうもの
偽陽性（FP）	ML モデルの予測で、モデルが誤って正のクラスを予測してしまうもの
フィーチャー	ML アルゴリズムの学習や ML モデルの予測に用いられる入力データの個々の測定可能な属性。
フィーチャーエンジニアリング	ML モデルに現れるべき基本的な関係性を最もよく表している生データの属性を，学習データとして使用するために特定する活動 (ISO/IEC TR 29119-11)
柔軟性	最初の仕様以外の状況でシステムが動作する能力（ISO/IEC TR 29119-11 以降
ファジィ・ロジック	0 と 1 の間の確実性係数で表される部分的な真実の概念に基づいた論理の一種
一般的な AI	認知能力の全範囲にわたって人間に匹敵する知的行動を示す AI（ISO/IEC TR 29119-11 類義語：強い AI
一般データ保護規則（GDPR）	EU および欧州経済地域の市民のデータに適用される、データ保護とプライバシーに関する欧州連合（EU）の規制。
グラフィカル・プロセッシング・ユニット（GPU）	ディスプレイへの出力を目的としたフレームバッファ内の画像生成を高速化するために、メモリを操作・変更するための特定用途向け集積回路。
グランドトゥルース	直接的な観察や測定によって得られる情報で、現実や真実であることがわかっているもの。
ハイパーパラメータ	ML モデルの学習を制御したり、ML モデルの設定を行うためのパラメータ

ハイパーパラメータチューニング	特定の目標に基づいて最適なハイパーパラメータを決定する活動
不適切なバイアス	特定のグループに悪影響を及ぼす結果をシステムにもたらすバイアスの一種。
インテリジェント・エージェント	観察と行動によって目標達成に向けて活動する自律的なプログラム。
クラスター間距離	異なるクラスタに属するデータポイントの類似性を測定する指標
解釈可能性	基礎となる AI 技術がどのように機能するかを理解しているレベル（ISO/IEC TR 29119-11
イントラクラスターメトリック	クラスター内のデータポイントの類似性を測定する指標
k-nearest neighbor	あるデータポイントのグループ・メンバーシップの可能性を、そのデータポイントに最も近いデータポイントのグループ・メンバーシップに依存して推定する分類のアプローチ。
学習アルゴリズム	学習データセットの特性に基づいて、ML モデルを生成するプログラム
LIME 方式	ML モデルの予測を説明するための「Local Interpretable Model-Agostic Explanations」プログラム
線形回帰	対象となる変数が数値の場合、観測されたデータに一次方程式を当てはめて変数間の関係をモデル化する統計手法
ロジスティック回帰	対象となる変数が数値ではなくカテゴリーである場合に、変数間の関係をモデル化する統計手法
機械学習（ML）	システムがデータや経験から学習することを可能にするために、計算技術を用いるプロセス（ISO/IEC TR 29119-11
平均二乗誤差（MSE）	推定値と実際の値の平均二乗差を示す統計的指標である
ML アルゴリズム	学習データセットから ML モデルを作成するためのアルゴリズム

ML ベンチマークスイート	ML モデルや ML アルゴリズムを様々な評価指標で比較するためのデータセット
ML フレームワーク	ML モデルの作成をサポートするツールやライブラリ
ML 機能	分類、回帰、クラスタリングなど、ML モデルによって実装される機能のこと。
ML モデルの評価	達成された ML 機能の性能指標を、要求された基準や他の ML モデルの性能と比較するプロセス
ML モデルのトレーニング	学習データセットに ML アルゴリズムを適用して ML モデルを作成するプロセス
ML モデルのチューニング	最適なパフォーマンスを得るためにハイパーパラメータをテストするプロセス
ML システム	1 つまたは複数の ML モデルを統合したシステム
ML のワークフロー	ML モデルの開発・展開を管理するための一連の活動のこと。
マルチエージェントシステム	複数のインテリジェントエージェントで構成されるシステム
ナローAI	特定の問題を解決するために、明確に定義された単一のタスクに焦点を当てた AI (ISO/IEC TR 29119-11) 類義語：弱い AI
自然言語処理（NLP	自然言語を読み、理解し、そこから意味を導き出す能力を提供するコンピューティングの分野
ニューラルネットワーク	プリミティブな処理要素を、重みを調整できるリンクで接続したネットワークで、各要素が入力値に非線形関数を適用して値を生成し、他の要素に伝達するか、出力値として提示するもの(ISO/IEC 2382) Synonym: 人工ニューラルネットワーク
ニューラルネットワーク・トロイの木馬	データポイズニング攻撃を用いてニューラルネットワークに注入された脆弱性を後から悪用することを目的としたもの。

ニューロモルフィック・プロセッサー	人間の脳の神経細胞を模倣して設計された集積回路
ニューロン	ニューラルネットワークのノードは、通常、複数の入力値を受け取り、活性化値を生成する
ノイズ	データの歪みや破損
非決定論的システム	特定の入力セットと開始状態を与えられても、常に同じ出力セットと最終状態を生成しないシステム
アウトライヤー	データ分布の全体的なパターンから外れている観測値
オーバーフィッティング	学習データとの対応が強すぎて、新しいデータへの汎化が困難な ML モデルが生成されること（ISO/IEC TR 29119-11 以降
パーセプトロン	1 つの層と 1 つのニューロンで構成されるニューラルネットワーク
精度	分類器を評価するために用いられる ML の機能的な性能指標で，予測された陽性が正しかった割合を測定するもの（ISO/IEC TR 29119-11 以降
学習済みモデル	入手した時点で既に学習済みの ML モデル
確率論的システム	挙動が確率で記述されるため、その出力を完全には予測できないシステム。
手続き的推論	ダイナミックな環境下で複雑なタスクを実行できるリアルタイム推論システムを構築するための AI 技術。
ランダムフォレスト	多数の決定木を構築して実行し、クラスの最頻値または個々の木の平均予測値を出力する、分類や回帰などの作業のためのアンサンブル ML 技術
推論技術	利用可能な情報から論理的手法を用いて結論を生成する AI（ISO/IEC TR 29119-11 以降
リコール	分類器の評価に用いられる ML の機能的な性能指標で，正しく予測された実際の陽性の割合を測定する（ISO/IEC TR 29119-11 以降

	類義語：感度
ROC（Receiver Operating Characteristic）曲線	識別しきい値を変化させたときの二値分類器の能力を示したグラフ。
リグレッション	与えられた入力に対して、数値または連続した出力値を得る ML 関数の一種（ISO/IEC TR 29119-11 以降
回帰モデル	与えられた数値入力に対する期待出力が連続変数である ML モデル（ISO/IEC DIS 23053 以降
強化学習	目的を達成するために試行錯誤を繰り返しながら ML モデルを構築する活動（ISO/IEC TR 29119-11 以降
報酬機能	強化学習の成功を定義する関数
リワードハッキング	知的エージェントが、本来の目的を達成することを犠牲にして、報酬関数を最大化するために行う活動（ISO/IEC TR 29119-11 以降
R-squared	データポイントがフィットした回帰線にどれだけ近いかを示す統計的指標。 Synonym: Coefficient of Determination
ルールエンジン	特定の条件が満たされたときに、どのようなアクションを起こすべきかを決定するルールの集合体
安全	定義された条件の下で、システムが人の生命、健康、財産または環境を危険にさらす状態にならないことを期待すること（ISO/IEC/IEEE 12207
サンプリングバイアス	データセットが、ML を適用したデータ空間を完全には代表していない場合のバイアスの一種
検索アルゴリズム	ゴールとなる状態や構造に到達するまで、可能性のあるすべての状態や構造のサブセットを系統的に訪れるアルゴリズム（ISO/IEC TR 29119-11 以降

自己学習型システム	試行錯誤による学習に基づいて動作を変化させる適応型システム（ISO/IEC TR 29119-11 以降）
シルエット係数	クラスタ間およびクラスタ内の平均差に基づく-1 から+1 の間のクラスタリング尺度 Synonym: シルエット・スコア
スーパーAI	人間の能力をはるかに超える人工知能ベースのシステム
監視下学習	入力データとそれに対応するラベルから ML モデルを学習する
サポートベクターマシン（SVM）	データポイントを超平面で区切られた多次元空間のベクトルと見なす ML 手法
テクノロジー・シンギュラリティ	技術の進歩が人の手では制御できなくなる未来の時点（ISO/IEC TR 29119-11 以降
テスト・オラクル問題	与えられたテスト入力と状態に対して、テストが合格か不合格かを判断するという課題がある
訓練データセット	ML モデルの学習に使われるデータセット
転移学習	学習済み ML モデルを、異なる関連タスクを実行するように修正する技術
透明度	AI ベースのシステムが使用するアルゴリズムやデータの可視性のレベル（ISO/IEC TR 29119-11 以降
真の陰性（TN）	否定的なクラスをモデルが正しく予測した場合の予測
真の陽性（TP）	モデルが正のクラスを正しく予測した場合の予測
アンダーフィッティング	学習データセットの基本的な傾向を反映していない ML モデルが生成され，その結果，正確な予測を行うことが困難なモデルとなること (ISO/IEC TR 29119-11)
教師なし学習	ラベルのないデータセットを用いて、入力データから ML モデルを学習する。

検証データセット	学習した ML モデルを評価するためのデータセットで、モデルのチューニングを目的としている
フォンノイマンアーキテクチャー	メモリ、中央処理装置、制御装置、入出力装置の 5 つの主要コンポーネントで構成されるコンピュータ・アーキテクチャ。
重み	ニューラルネットワークにおけるニューロン間の接続の内部変数で、出力の計算方法に影響を与え、ニューラルネットワークの学習に伴って変化する。