情報処理学会論文誌 Vol.63 No.12 1830–1839 (Dec. 2022)

XGBoost法により学習させたAIモデルの振舞いの形式検証

來間 啓伸[1,a)] 明神 智之[1] 佐藤 直人[1] 小川 秀人[1]

受付日 2022年2月10日, 採録日 2022年9月2日

概要：論理的に記述し分析することが困難な事象を処理するソフトウェアを構成する方法として，機械学習への期待が高まっている．一方，機械学習によって作成されたAIモデルの振舞いを予測することは困難であり，明らかに不適切な推論結果が出力される可能性が否定できない．AIモデルの振舞いの形式検証は，指定した入力データ範囲の中で推論出力データが検証性質を満たすことを網羅的に検証する点で，この課題の解決に有効なアプローチである．その反面，形式検証には多くの計算が必要であり，現実的な計算資源の下では検証結果が得られない場合もある．本稿では，XGBoost法を使って学習させた決定木集合モデルの振舞いの形式検証手法を提案し，検証の計算量と精度を制御するためのモデルの縮約について述べる．米国の住宅価格データベースを学習データとして作成した決定木集合モデルを題材に，ツールを使ったケーススタディを行い，提案手法のフィージビリティを評価した．大規模AIモデルへの適用を通じて手法を洗練することが，今後の課題である．

キーワード：AIソフトウェアのテスト，機械学習，決定木集合モデル，プログラム検証，充足可能性問題

Formal Verification of AI Model Bahaviors Trained by XGBoost Machine-learning Method

Hironobu Kuruma[1,a)] Tomoyuki Myojin[1] Naoto Sato[1] Hideto Ogawa[1]

Received: February 10, 2022, Accepted: September 2, 2022

Abstract: Machine learning is expected to be a development method of software, which process phenomena that are difficult to describe and analyze logically. Since machine learning generates AI models inductively, their behaviors are unpredictable. An AI model may output data which is obviously inconsistent with human knowledge. Formal verification of AI models solves this problem by checking that the output of the model satisfies the given property for every input specified in the precondition. However, much computing resources are needed for formal verification and frequently verification does not terminate in a practical time. In this paper, we propose a formal verification method for decision tree ensemble models trained using XGBoost machine learning method and describe a model reduction technique for controlling both verification time and verification accuracy. We evaluated the feasibility of our method through a verification case study of house price prediction model trained on a house sales dataset. Our feature work is to apply the proposed method to larger models and refine it.

Keywords: testing of AI software, machine learning, decision tree ensemble, program verification, satisfiability problem

1. はじめに

実世界からのデータ収集と実世界への作用が可能な情報処理装置がネットワークを通じて広く連携することで，計算機システムが実世界と密接に結び付いて動作するCyber

[1] 株式会社日立製作所研究開発グループ
Research and Development Group, Hitachi, Ltd., Yokohama, Kanagawa 244–0817, Japan
[a)] hironobu.kuruma.zg@hitachi.com

Physical Systems（CPS）が実現されつつある．ここで，実世界は論理的に分析し記述することが困難な不確実性を含んでおり，これまでの計算機システムは不確実性を捨象して論理的に記述できる範囲の処理を担ってきた．しかし，CPS では実世界との結び付きが密になるので，計算機システムにも不確実な事象を処理することが求められる [5]．機械学習は，実世界の不確実性に適応するソフトウェアを，データからの学習により機械的に作成する開発手法として期待されている．機械学習では，あらかじめ用意した学習データセットを使って AI モデルを訓練する．運用時には，学習データとは異なるデータを入力して AI モデルに推論させ，推論出力データを得る．AI モデルは学習データセットからデータの特徴を学習し，入力されたデータから特徴を抽出して推論するので，画像認識のように対象の特徴を定義し処理方法を論理的に記述することが困難な事象を処理するのに適している．

AI モデルをシステムに組み込む際には，設計した入力データ範囲の中で，どのような入力データに対して不適切な推論結果が出力されるのか，どのような入力データ範囲内であれば妥当な推論結果が期待できるのか，を把握しておくことが重要になる．AI モデルが不適切な推論を行う入力データが特定できれば学習にフィードバックできるうえ，AI モデルを分析して誤りを修正できる可能性もある．また，推論結果が妥当であると思われる入力データ範囲内で AI モデルを使い，それ以外の入力データに対しては他の手段を使うようにシステムを設計することもできる．しかし，対象を分析して仕様を記述し，仕様を満たすソフトウェアを実装する演繹的なソフトウェア開発手法とは異なり，機械学習は学習データに基づく帰納的なソフトウェア開発手法 [12] であるため，AI モデルが満たすべき仕様が存在しない．したがって，仕様を満たすことを検証する演繹的なテスト・検証手法は，AI モデルのテスト・検証には適用できない [13]．機械学習の過程では，学習データセットから取り分けた試験データセットをテストオラクルとして推論精度を評価するが，学習データセットは AI モデルに入力されうるデータのサンプルにすぎないので，ソフトウェアのテストと比較するとデータ数の点でもカバレッジの点でも不十分である．

この課題に対して本稿では，AI モデルの振舞いの形式検証の視点からアプローチする．形式検証では，入力データの範囲と期待される推論出力データの条件を指定し，範囲内のすべての入力データに対して AI モデルの推論出力データが条件を満たすことを論理的に検証する．以下では，入力データの範囲を前提条件，推論出力データの条件を検証性質と呼ぶ．演繹的な開発では前提条件と検証性質は仕様から導かれるが，AI モデルの検証では人の形式知や倫理的な制約，システム内の他のコンポーネントとのインタフェースから要請される部分仕様などから設定する．AI モデルの性質上，任意の入力データに対する正しい推論出力データは不明なので，本稿の形式検証は AI モデルの推論の正しさの検証ではない．AI モデルの推論出力データが，人が納得できるないし設計上許容できる範囲内にあることを検証する．推論出力データが検証性質を満たさない具体例は，反例（counterexample）と呼ばれる．反例は，推論出力データが誤っていることを直接示すわけではないが，人が表明した期待とは異なる点で精査すべき具体例である．

本稿では，機械学習法の 1 つである XGBoost [1] を使って学習させた回帰問題を解く AI モデルを対象とした，振舞いの形式検証手法を提案する．簡単のため，XGBoost を使って学習させた AI モデルを XGBoost モデルと呼ぶ．XGBoost モデルは多数の独立した決定木から構成される決定木集合（decision tree ensemble）モデルであり，振舞いの形式検証では，これらの決定木のパスの組合せについて検証性質が満たされることを確認する必要がある．一般に組合せ数は膨大になるので，検証のための XGBoost モデルの縮約方法を示す．

以下，2 章では XGBoost モデルを論理式に変換して形式検証する方法について述べるとともに，検証効率を向上させるためのモデルの縮約方法を提案する．3 章では，縮約による検証時間削減の効果を，ツールを使ったケーススタディにより示す．4 章では，アプローチの有効性と検証手法の効果について検討する．関連研究を 5 章にあげ，6 章で本稿をまとめる．

2. XGBoost モデルの形式検証

2.1 XGBoost モデル

XGBoost モデルは決定木をベースとする AI モデルであり，多量のデータを処理できることから広く使われている．1 つの XGBoost モデルは，一般に多数の決定木から構成される．簡単のため，2 つの決定木から構成されたモデルの例を**図 1** に示す．ここで，図の○は決定木のノード，△はリーフ，矢印はエッジを表す．エッジには方向性があり，矢印の根元を始点，指す先を終点とする．エッジの始点には必ず 1 つのノードが存在し，エッジの終点には必ず 1 つのノードまたはリーフが存在する．決定木は非循環的であり，エッジを始点から終点へとノードを介して連鎖的にた

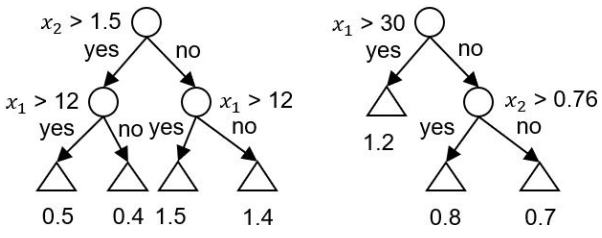

図 1 決定木集合モデルの例
Fig. 1 Decision tree ensemble model.

どっても，同じノードに戻ることはない．ルートノードはエッジの始点にはなるが終点にはならないノードで，1 つの決定木に 1 つだけ存在する．

決定木の各ノードには入力データに関する決定条件が対応付いており，入力データが決定条件を満たす場合と満たさない場合で yes または no のラベルのエッジが選択され，終点のノードに進む．このとき，決定条件を満たすか満たさないかはあいまい性なく判定でき，必ず 1 つのエッジが選択されるものとする．エッジの終点のノードでも同様に決定条件による振り分けが行われるので，ルートノードから始めて，入力データに従って振り分けを行うと，必ず 1 つのリーフに到達する．

リーフにはウェイトが対応付けられており，入力データに対して選ばれたリーフに対応付けられているウェイトが，その決定木の推論出力データである．図 1 では，リーフの下の数字がウェイトを表す．XGBoost モデルの推論出力データは，このようにして選ばれたウェイトをすべての決定木について足し合わせた値である．図 1 のモデルでは，たとえば x_1 の値が 10，x_2 の値が 1 の入力データに対して，左側の決定木は右端のリーフに到達してウェイトは 1.4，右側の決定木は中央のリーフに到達してウェイトは 0.8 なので，$x_1 = 10$，$x_2 = 1$ に対するモデルの推論出力データは 2.2 となる．

2.2 形式検証

AI モデルの振舞いの形式検証では，任意の入力データ x に対する AI モデルの推論出力データが y のとき，前提条件 $P(x)$ と検証性質 $I(y)$ の間に

$$P(x) \implies I(y)$$

が成り立つことを検証する．具体的には，AI モデルを入力データ x と推論出力データ y の関係を表す論理式 $R(x, y)$ に変換し，前提条件と検証性質の否定を論理式で結合した

$$R(x, y) \wedge P(x) \wedge \neg I(y) \tag{1}$$

が真になる x と y の値を，充足可能性問題として SMT（Satisfiability Modulo Theories）ソルバ [18] を使って機械的に探索する．そのような値が見つかれば，それは前提条件を満たし検証性質を満たさないので反例である．逆に，真になる値が見つからなければ，前提条件の範囲内のすべての入力データについて，推論出力データが検証性質を満たすことがいえる．

以下，XGBoost モデルから論理式への変換について述べる．XGBoost モデルの入力データを x，モデルを構成する決定木のノード m に対応付けられた決定条件を $dc_m(x)$ とすると，ノード m を始点とするエッジ $\langle m, n\rangle$ が選ばれる条件 $e_{m,n}(x)$ は，

$$e_{m,n}(x) = \begin{cases} dc_m(x) & \langle m, n\rangle \text{ のラベルが yes のとき} \\ \neg dc_m(x) & \langle m, n\rangle \text{ のラベルが no のとき} \end{cases}$$

である．これをエッジの列であるパスに拡張すると，決定木 k でルートノード $n_0^{(k)}$ からリーフ $l_i^{(k)}$ に到達するパス $q_i^{(k)}$ が選ばれる条件 $q_i^{(k)}(x)$ は，

$$e_{n_0{}^{(k)}, n_{i_1}^{(k)}}(x) \wedge e_{n_{i_1}{}^{(k)}, n_{i_2}^{(k)}}(x) \wedge \cdots \wedge e_{n_{i_m}{}^{(k)}, l_i^{(k)}}(x)$$

と表せる．ここで，$n_{i_1}^{(k)}$ から $n_{i_m}^{(k)}$ は $n_0^{(k)}$ から $l_i^{(k)}$ の間にあるノードである．リーフ $l_i^{(k)}$ に対応付いているウェイトを $w_i^{(k)}$ とすると，パス i についての入力データ x と決定木 k の推論出力データ $y^{(k)}$ の関係 $R_i^{(k)}$ は，

$$\begin{aligned} R_i^{(k)}(x, y^{(k)}) &\Leftrightarrow q_i^{(k)}(x) \wedge y^{(k)} = w_i^{(k)} \\ &\Leftrightarrow \left(\bigwedge_{j=0}^{m} e_{n_{i_j}^{(k)}, n_{i_{j+1}}^{(k)}}(x) \right) \wedge y^{(k)} = w_i^{(k)} \end{aligned}$$

である．ここで，$\bigwedge_{j=0}^{m}$ は $\wedge$ の後ろの式の j が 0 から m までの論理積を表し，$n_{i_0}^{(k)}$ はルートノード $n_0^{(k)}$，$n_{i_{m+1}}^{(k)}$ はリーフ $l_i^{(k)}$ とする．入力データ x に対してパス $q_i^{(k)}$ が選ばれるとき，つまり $q_i^{(k)}(x)$ が真のとき，$y^{(k)}$ の値が $w_i^{(k)}$ であれば $R_i^{(k)}(x, y^{(k)})$ は真になる．一方，入力データ x に対してパス $q_i^{(k)}$ が選ばれないとき，$R_i^{(k)}(x, y^{(k)})$ は偽になる．

決定木では，入力データが与えられたときに，ただ 1 つのパスが選ばれて推論出力データが決まる．決定木 k について入力データ x に推論出力データ $y^{(k)}$ を対応付ける論理式 $R^{(k)}$ は，リーフに到達するすべてのパスについての下記の論理和である．

$$R^{(k)}(x, y^{(k)}) \Leftrightarrow \bigvee_{i=1}^{L^{(k)}} R_i^{(k)}(x, y^{(k)}) \tag{2}$$

ここで，$L^{(k)}$ は決定木 k のパスの数である．パスは必ず 1 つだけ選ばれるので，1 つの入力データに対して $R_i^{(k)}(x, y^{(k)})$ が真になる $y^{(k)}$ の値が 1 つ決まる．XGBoost モデル全体では，各決定木の推論出力データを足し合わせて

$$R(x, y) \Leftrightarrow \left(\bigwedge_{k=1}^{K} R^{(k)}(x, y^{(k)}) \right) \wedge y = \sum_{k=1}^{K} y^{(k)}$$

と置き換える．ここで，K は決定木の数，y は XGBoost モデルの推論出力データである．このようにして，たとえば図 1 のモデルは，次の論理式に変換することができる．

$$\begin{aligned} &((x_2 > 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 0.5)\vee \\ &(x_2 > 1.5) \wedge (x_1 \leq 12) \wedge (y^{(1)} = 0.4)\vee \\ &(x_2 \leq 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 1.5)\vee \\ &(x_2 \leq 1.5) \wedge (x_1 \leq 12) \wedge (y^{(1)} = 1.4))\wedge \\ &((x_1 > 30) \wedge (y^{(2)} = 1.2)\vee \end{aligned}$$

$$(x_1 \leq 30) \wedge (x_2 > 0.76) \wedge (y^{(2)} = 0.8) \vee$$
$$(x_1 \leq 30) \wedge (x_2 \leq 0.76) \wedge (y^{(2)} = 0.7)) \wedge$$
$$(y = y^{(1)} + y^{(2)})$$

XGBoost モデルを論理式に変換し，充足可能性問題として検証することで，指定した範囲の入力データについて推論出力データが検証性質を満たすことが論理的に結論できる，ないしは検証性質を満たさない反例が得られる．このことは，個々に与えたテストデータについて推論出力データが検証性質を満たすことを検証するテストと比べたときの，形式検証の利点である．その半面，検証時の計算量が多く，検証できるモデルの大きさが制約されることが課題である．充足可能性を探索する式 (1) は，

$$R(x,y) \wedge P(x) \wedge \neg I(y) \Leftrightarrow$$
$$\bigwedge_{k=1}^{K} R^{(k)}(x, y^{(k)}) \wedge \left(y = \sum_{k=1}^{K} y^{(k)} \right) \wedge P(x) \wedge \neg I(y) \Leftrightarrow$$
$$\bigwedge_{k=1}^{K} \left(\bigvee_{i=1}^{L^{(k)}} (q_i^{(k)}(x) \wedge y^{(k)} = w_i^{(k)}) \right) \wedge$$
$$\left(y = \sum_{k=1}^{K} y^{(k)} \right) \wedge P(x) \wedge \neg I(y)$$

と展開することができ，最悪の場合にはすべての決定木 k（$1 \leq k \leq K$）についてパス i（$1 \leq i \leq L^{(k)}$）の組合せの充足可能性を探索する必要があることを示している．$L^{(k)}$ の概数を L とすると，パスの組合せの総数は

$$L^K$$

となることから，最悪の場合の計算量は決定木数の指数関数になる．

2.3 モデルの縮約

本稿では，充足可能性探索の計算量を削減して検証効率を高めるために，以下のアプローチをとる．

- 冗長なパスの削除
  前提条件を満たさないパスを，充足可能性探索式から削除する．
- 推論出力データへの寄与が小さい決定木の近似
  ウェイトの最大値と最小値の差が小さい決定木を充足可能性探索式から削除し，削除した決定木の推論出力データをウェイトの最大値と最小値で置き換える．

上記の充足可能性探索式は，さらに

$$R(x,y) \wedge P(x) \wedge \neg I(y) \Leftrightarrow$$
$$\bigwedge_{k=1}^{K} \left(\bigvee_{i=1}^{L^{(k)}} (q_i^{(k)}(x) \wedge P(x) \wedge y^{(k)} = w_i^{(k)}) \right) \wedge$$
$$\left(y = \sum_{k=1}^{K} y^{(k)} \right) \wedge \neg I(y)$$

と変形することができ，$q_i^{(k)}(x) \wedge P(x)$ を満たす入力データ x が存在しないパス i についてはパスの論理式

$$q_i^{(k)}(x) \wedge y^{(k)} = w_i^{(k)}$$

をあらかじめ削除しておいても充足可能性探索に影響しないことが分かる．前提条件を満たさないパスの削除は，次の手順で行う．

(1) 決定木をパスに分割してパスの論理式 $q_i^{(k)}(x)$ を作成．
(2) $q_i^{(k)}(x) \wedge P(x)$ の充足性を，SMT ソルバを使って検査．
(3) 充足不能なパスの論理式を除去．

冗長なパスの削除に要する計算量はパスの総数に比例し，$O(L \times K)$ のオーダである．

決定木の近似では，ウェイトの最大値と最小値の差が閾値以下の決定木 k について

$$\bigvee_{i=1}^{L(k)} \left(q_i^{(k)}(x) \wedge y^{(k)} = w_i^{(k)} \right)$$

を

$$(w_i^{(k)} \text{ の最小値}) \leq y^{(k)} \wedge y^{(k)} \leq (w_i^{(k)} \text{ の最大値}) \quad (3)$$

で置き換える．これは推論出力データの値範囲を広く見積もることになるので，反例が見つからなかった場合には推論出力データが検証性質を満たすことを保証できる．一方，反例が見つかったとしても，真の推論出力データが検証性質に反するとは限らない．すなわち，近似による誤差の範囲内で，真の値とは異なる値が反例として検出される可能性がある．本稿では，真の値が検証性質を満たすにもかかわらず決定木を近似した検証では満たさないと判定された反例を，偽反例と呼ぶ．決定木を近似したモデルで反例が検出されたとき，真の反例か偽反例かの区別は，検出された反例を近似前のモデルに入力し推論出力データが検証性質を満たすかどうかを調べることによって，容易に行うことができる．

近似は決定木を単位に行うが，一般に複数の決定木が近似対象になりうる．このため，次の手順で近似処理を行う．

(1) ウェイトの最大値と最小値の差が閾値以下の決定木を抽出．
(2) 抽出した決定木の推論出力データの和 y' を，ウェイトの最大値の和と最小値の和で制約．

$$\text{最小値の和} \leq y' \wedge y' \leq \text{最大値の和}$$

(3) 残りの決定木について充足可能性探索式を作り，モデル全体の推論出力データ y の値を残りの決定木の推論出力データ $y^{(k')}$ の和と y' で置き換え．

$$y = \sum y^{(k')} + y'$$

近似する決定木の数を h とすると，充足可能性探索の計

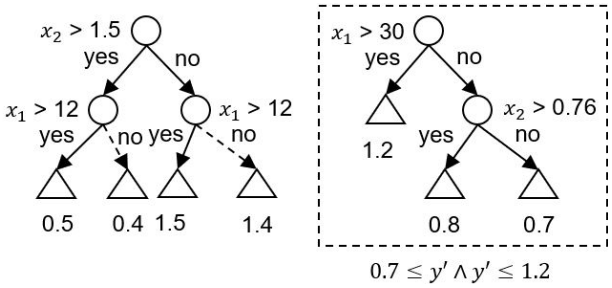

図 2 縮約した決定木集合モデル
Fig. 2 Reduced decision tree ensemble model.

算量のオーダは最悪の場合でも $O(L^{K-h})$ となる．これに対し，近似対象とする決定木の抽出に要する計算量のオーダは，$O(L \times K)$ である．一方，近似による推論出力データの誤差は，最大で

近似閾値 × 近似する決定木の数

である．誤差による偽反例が検出される可能性があるため，決定木を近似した場合の検証結果は次のように解釈できる．

- 反例が検出されないとき，検証性質が満たされる．
- 真の反例が検出されたとき，検証性質は満たされない．
- 偽反例のみが検出されたとき，判定できない．

有用性を損なわずに近似できる決定木の数はモデルと検証したい性質に依存するので，問題ごとに近似の有効性を考慮する必要がある．

冗長なパスの削除と決定木の近似の例を，**図 2** に示す．いま，検証時に与えた前提条件が $x_1 > 15$ であるとすると，左側の決定木に破線で示した，決定条件 $x_1 > 12$ に対して no のラベルのエッジを含むパスは前提条件を満たさない．すなわち，前提条件で規定された入力データ範囲には，これらのパスを選択する入力データは存在しないので，決定木を論理式に変換する際に除外しておくことができる．また，検証時に与えた近似閾値が 0.5 であるとすると，右側の決定木はウェイトの最大値 1.2 と最小値 0.7 の差が近似閾値以下なので，近似対象として抽出できる．このとき，入力データに対してどのリーフが選ばれてもウェイトは 0.7 と 1.2 の間にあるので，右側の決定木の推論出力データを y' として決定木の論理式を

$$0.7 \leq y' \wedge y' \leq 1.2$$

で置き換える．図 2 の縮約したモデルを変換すると，次の論理式が得られる．

$$\begin{aligned}&((x_2 > 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 0.5) \vee \\ &(x_2 \leq 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 1.5)) \wedge \\ &(0.7 \leq y' \wedge y' \leq 1.2) \wedge \\ &(y = y^{(1)} + y')\end{aligned}$$

3. ケーススタディ

3.1 検証対象モデル

2 章で述べた検証手法を評価するため，米国の住宅価格データベース [19] を学習データとして XGBoost 法で学習させた住宅価格予想モデル [15] をサンプルに，検証ツールを使った検証実験を行った．このモデルは，住宅を特徴づける

- リビングルームの面積（290 から 13,540 平方フィート）
- 寝室の数（0 から 33）
- 土地面積（520 から 1,651,359 平方フィート）
- 建物の状態（1 から 5 までの 5 段階）
- 構造とデザインのグレード（1 から 13 までの 13 段階）
- 地上階の面積（290 から 9,410 平方フィート）
- 地下室の面積（0 から 4,820 平方フィート）

の 7 つのデータから，

- 住宅価格（ドル）

を予想する．サンプルとした XGBoost モデルは，リーフの数が 8 の決定木 80 個，7 の決定木 19 個，6 の決定木 1 個の合計 100 個の決定木からなり，パスの組合せの総数は約 $8^{100} \approx 2 \times 10^{90}$ である．

ケーススタディでは，次の 3 つの検証内容について形式検証を行った．

- 検証内容 1
  前提条件：すべての住宅
  検証性質：住宅価格は 10,000,000 ドル未満
- 検証内容 2
  前提条件：すべての住宅
  検証性質：住宅価格は 100,000 ドル以上
- 検証内容 3
  前提条件：リビングルームの面積が 7,000 平方フィート以上であり構造とデザインのグレードが 12 以上
  検証性質：住宅価格は 500,000 ドル以上

検証内容 1 はモデルの推論出力データが不当に高い価格にならないことの検証を，検証内容 2 は推論出力データが不当に安い価格にならないことの検証を意図している．検証性質 3 は，高級な住宅について妥当な価格を予想することを検証するものである．

3.2 検証実験

上記の検証内容は，次のように形式化することができる．

- 検証内容 1

$$true \implies y < 10000000$$

- 検証内容 2

$$true \implies y \geq 100000$$

- 検証内容 3

$$x_1 \geq 7000 \wedge x_5 \geq 12 \implies y \geq 500000$$

ここで，変数 y は住宅価格の推論出力データ，変数 x_1 と x_5 はそれぞれリビングルームの面積，構造とデザインのグレードの入力データを表す．

検証内容 1 と 2 は前提条件 $P(x)$ が入力データ x によらず真であり，すべてのパスについて $q_i^{(k)}(x) \wedge P(x) \Leftrightarrow q_i^{(k)}(x)$ が成り立つ．すなわち冗長なパスが存在せず，冗長なパスの削除はモデルの縮約に寄与しない．一方，検証内容 3 では前提条件の真偽が入力データに依存するので，冗長なパスの削除が適用できる．これに対し決定木の近似は，近似される決定木が XGBoost モデルの構造によって決まり，検証内容に依存しない．このとき，偽反例が出現する近似閾値が検証内容に依存する．

検証実験では，検証内容 1 から 3 について近似閾値を変えて決定木の近似を行い，モデルの縮約処理を行わない検証ツールと検証時間を比較した．検証内容 3 については，冗長なパスを削除した場合の検証時間も測定した．以下の表で，「従来ツール」はモデルの縮約処理を行わない文献 [15] の検証ツールを使った場合の検証時間を表す．決定木数は検証対象となる決定木の数であり，サンプルとした XGBoost モデルの決定木数 100 から近似により除去した決定木数を引いた値である．パス数は，モデルのパスの総数 779 から，決定木の近似およびパスの削除により除かれたパスの数を引いた値である．近似誤差は，近似される決定木のウェイトの最大値の和からウェイトの最小値の和を引いた値で，決定木を近似したことにより生じる推論出力データの誤差の最大値を表す．検証時間の測定は同一条件で 5 回行い，平均値を検証時間とした．検証に用いた機器の CPU は Intel Core i7-10700K 3.8 GHz，メモリは 32 GB，OS は Windows 10 Pro 64 bit である[*1]．

検証内容 1 の検証結果を，**表 1** に示す．サンプルとした XGBoost モデルでは，近似閾値が 0 のときには近似される決定木はなかった．一般に，ウェイトの最大値が最小値と等しい決定木がモデルの中に存在するとき，誤差なく定数と置き換えて検証効率を上げることができる．3.3 節に示すように縮約処理に要する時間は短いので，近似閾値が 0 のときの従来ツールとの検証時間の差は，変換後の論理式の構造の違いによる SMT ソルバの処理効率の差を表していると考えられる．

検証内容 1 については近似閾値が 150,000 以下では反例が存在せず，すべての住宅について予想価格が 10,000,000 ドル未満であることが検証できた．前述のように，本稿の近似方法では推論出力データの値範囲を広く見積もるので，反例および偽反例が見つからなかった場合には推論出力データが検証性質を満たすことが保証でき，近似閾値が 150,000 でも十分な検証精度があるといえる．一方，近似閾値が 160,000 のときには偽反例が検出されており，検証精度が不十分なので近似閾値を小さくして検証する必要があることが分かる．

[*1] Intel は，アメリカ合衆国およびその他の国における Intel Corporation の登録商標です．Windows は，アメリカ合衆国およびその他の国における Microsoft Corporation の登録商標です．

表 1 検証内容 1 の検証時間の比較
Table 1 Comparison of time for verifying property 1.

近似閾値	決定木数	パス数	近似誤差	検証時間（sec）
0	100	779	0	2,860.36
20,000	99	771	17,344	2,204.92
120,000	69	536	2,349,419	259.65
150,000	60	468	3,580,569	100.93
160,000	59	461	3,738,532	2.22（偽反例）
従来ツール	100	779	-	2,295.94

表 2 検証内容 2 の検証時間の比較
Table 2 Comparison of time for verifying property 2.

近似閾値	決定木数	パス数	近似誤差	検証時間（sec）
20,000	99	771	17,344	0.48（反例）
60,000	91	707	387,504	0.51（反例）
80,000	84	653	891,273	0.46（反例）
100,000	74	575	1,795,270	0.42（偽反例）
従来ツール	100	779	-	0.42（反例）

近似閾値を大きくして充足可能性探索の対象とする決定木を少なくするにつれて検証時間が短くなる傾向がみてとれ，近似閾値が 150,000 のときの検証時間は従来ツールの約 1/23 である．検証時間には縮約処理や前述の論理式の構造の違いに起因するオーバヘッドが含まれており，モデルの縮約により検証効率が向上したといえる．

検証内容 2 については，予測価格が 100,000 ドル未満になりうることを示す反例が検出された．ツールは反例ないし偽反例が 1 つ検出された時点で検証を終了するため，**表 2** に示すように，反例が存在しない場合に比べて検証時間は短い．また，論理式の複雑さよりも充足可能性を探索する順序に依存するので，近似する決定木が多いほど検証時間が短くなるとは限らない．

検証性質の 100,000 ドルと比較して近似誤差の影響が無視できないと考えられるが，近似閾値 80,000 まで正しく反例が検出された．反例の 1 つを，下記に示す．

- リビングルームの面積が 11,105 平方フィート
- 寝室の数が 7
- 土地面積が 167,488 平方フィート
- 建物の状態が 4
- 構造とデザインのグレードが 5
- 地上階の面積が 5,385 平方フィート
- 地下室の面積が 75 平方フィート
- 住宅価格が −223,680 ドル

住宅価格が負値になっており，期待される推論結果とは明らかに異なる．

表 3 検証内容 3 の検証時間の比較
Table 3 Comparison of time for verifying property 3.

近似閾値	決定木数	パス数	近似誤差	検証時間（sec）
40,000	97	755	93,814	45.91
60,000	91	707	387,504	21.63
80,000	84	653	891,273	13.48
90,000	80	621	123,854	0.79（偽反例）
冗長パス削除	100	501	0	36.74
従来ツール	100	779	-	38.46

表 4 決定木近似後に冗長パスを削除した際の検証内容 3 の検証時間
Table 4 Verification time of property 3 through approach 1.

近似閾値	決定木数	パス数	近似誤差	検証時間（sec）
40,000	97	483	93,814	34.89
60,000	91	455	387,504	21.01
80,000	84	417	891,273	14.25
90,000	80	397	123,854	0.68（偽反例）

検証内容 3 の形式検証では，**表 3** に示すように近似閾値が 90,000 の場合を除いて反例が検出されず，80,000 以下のどの近似閾値を用いても検証性質が満たされることがいえる．一方，近似閾値が 90,000 のときは検証精度が不十分であり，偽反例が検出されている．近似閾値が 80,000 のときの検証時間は，従来ツールの約 1/3 であった．

前述のように，検証内容 3 には冗長なパスの削除によるモデルの縮約が適用できる．冗長なパスの削除のみを適用したときの検証結果を，表 3 の「冗長パス削除」に示す．冗長なパスの削除によりモデルのパスの数が 779 から 501 に減少し，検証時間は従来ツールよりも短くなっている．

決定木の近似と冗長なパスの削除を組み合わせてモデルを縮約する際に，次の 2 つのアプローチが考えられる．

- アプローチ 1
  近似閾値に従って決定木を近似した後に，冗長なパスを削除する．
- アプローチ 2
  冗長なパスを削除した後に，近似閾値に従って決定木を近似する．

アプローチ 1 により検証内容 3 を検証した際の検証時間を，**表 4** に示す．先に決定木を近似するので，近似閾値に対する決定木数は決定木の近似のみを行う場合と同じであるが，その後に冗長なパスを削除するためモデル中のパス数は減少する．冗長なパスの削除は検証に誤差を生じないので，近似誤差は削除前と変わらない．決定木の近似により決定木ごと除去されるパスの数と，その後に冗長なパスとして削除されるパスの数を，**表 5** に示す．決定木の近似により除去されるパスの数は，決定木の近似のみ行う場合と同じである．一方，近似後の決定木から削除される冗長なパスの数は，近似される決定木が増えるにつれて少なくなる．この結果，除去されるパスと合わせた削除パス合計は，近似閾値とともにゆるやかに増加する．

表 5 決定木近似後に冗長パスを削除した際のパス削除数の内訳
Table 5 Number of reduced paths through approach 1.

近似閾値	近似による除去	冗長パス削除	削除パス合計
40,000	24	272	296
60,000	72	252	324
80,000	126	236	362
90,000	158	224	382

表 6 冗長パス削除後に決定木を近似した際の検証内容 3 の検証時間
Table 6 Verification time of property 3 through approach 2.

近似閾値	決定木数	パス数	近似誤差	検証時間（sec）
40,000	88	459	261,343	26.80
60,000	80	422	648,751	14.42
70,000	78	414	775,839	13.82
80,000	71	373	1,307,918	0.88（偽反例）

表 7 冗長パス削除後に決定木近似した際のパス削除数の内訳
Table 7 Number of reduced paths through approach 2.

近似閾値	冗長パス削除	近似による除去	削除パス合計
40,000	278	42	320
60,000	278	79	357
70,000	278	87	365
80,000	278	128	406

表 6 は，アプローチ 2 による検証時間である．はじめにパスを削除することで決定木のウェイトの最小値と最大値の差が縮まり，近似閾値に対して近似される決定木が増加する．この結果，低い近似閾値から検証時間の短縮効果があるが，近似閾値が 80,000 の段階で偽反例が出現した．冗長なパスの削除により削除されるパスの数と，その後に行う決定木の近似により決定木ごと除去されるパスの数を，**表 7** に示す．冗長なパスとして削除されるパスの数は，冗長なパスの削除のみ行う場合と同じである．はじめに冗長なパスを削除するので，近似される決定木はモデルの構造だけでなく前提条件にも依存する．1 つの決定木中のパスの数は冗長なパスの削除により減少しているにもかかわらず，近似閾値に対して近似される決定木が増えたことから，決定木の近似により除去されるパスの数は決定木の近似のみ行う場合より多かった．

いずれのアプローチでも，冗長なパスを削除することで近似閾値に対して近似される決定木が増加し，低い近似閾値から検証時間の短縮効果が得られる．冗長なパスは前提条件によって定まり，近似閾値によらないため，近似閾値が小さく決定木数が多いときには削除の効果が大きく検証時間が短くなる．その反面，決定木数が少なくなるにつれて効果が小さくなり，近似のみの場合と検証時間が変わらなくなる．実用上は，検証対象モデルが大きく試行による近似閾値の決定が難しいとき，小さい近似閾値で検証時間の短縮効果が得られる点で，冗長なパスの削除は有効で

ある．

3.3 縮約処理のオーバヘッド

3.2 節に示した検証時間には，サンプルとした XGBoost モデルからの縮約処理と充足可能性探索の時間が含まれている．このうちの縮約処理に要した時間の計測結果を，以下に示す．

2.3 節で述べた決定木の近似処理は，ウェイトの最大値と最小値を得るため決定木をトラバースする処理が中心であり，近似閾値や検証内容によらずすべての決定木をトラバースするので，計算量はモデル中のパスの数に依存する．検証実験では検証内容 1 から 3 の各々の近似閾値について 100 個の決定木の 779 個のパスをトラバースしており，近似処理にかかった時間は約 0.011 秒から 0.018 秒であった．

一方，冗長なパスの削除では SMT ソルバがパスの数だけ呼び出されるが，前提条件とパスの間の充足可能性を探索する際の論理式は単純であり，処理時間は短い．検証内容 3 で冗長なパスの削除のみを適用した場合には，779 個のパスについて前提条件との充足可能性を探索し，その処理時間は約 0.134 秒であった．

検証内容 3 の検証で決定木の近似と冗長なパスの削除を組み合わせる場合，決定木の近似を先に行うアプローチ 1 では，近似処理の時間は約 0.012 秒，パスの削除は近似後の決定木に対して行うため近似閾値 40,000，60,000，80,000 に対してそれぞれ 0.126 秒，0.117 秒，0.112 秒と順に短くなっている．アプローチ 2 では，パスの削除を先に行うため近似処理の時間は少し短くなり約 0.009 秒，パスの削除処理の時間は約 0.132 秒であった．

4. 議論

AI モデルを機械学習によって生成されたプログラムと考えると，その振舞いの形式検証はプログラム検証 [6] と見なすことができ，システム開発における両者の位置づけは共通する [10]．ここで，プログラム検証ではプログラムが満たすべき仕様が明示的に与えられ，仕様を満たすことが検証できたプログラムは「正しい」と判定できるのに対し，AI モデルの振舞いの形式検証では検証性質をアドホックに設定するので，検証性質を満たすことが検証できた AI モデルは「もっともらしい」と判定できる．AI モデルはプログラムのような複雑な制御構造を持たないデータの集まりであり，データ量の多さに留意すれば，機械的に変換して網羅的に自動検証することは難しくない [7], [9], [14]．このような特徴から，システムに不整合を起こさないための条件を検証性質として設定することで，不適切な推論出力データを反例として機械的に検出できる．これらを考慮すると，AI モデルの振舞いの形式検証は，AI を利用するシステムの開発において有効な検証アプローチであるといえる．

本稿で提案した XGBoost モデルの検証手法では，モデルを縮約することで充足可能性探索の計算量を削減する．2.3 節のモデル縮約アプローチのうち，冗長なパスの削除は，入力データの範囲を制約する前提条件が存在するときに，有効である．決定木の近似では，近似閾値によって近似される決定木はモデルから定まり，偽反例が現れる近似閾値が検証内容に依存する．このとき，ウェイトの最大値と最小値の差が近接する決定木がモデル内に多く含まれている場合には，近似閾値のわずかな差で多数の決定木が近似され，近似閾値の調整による偽反例の回避が難しくなる．通常の方法で構成された XGBoost モデルであれば，このような決定木が多数生成されることはないと考えられるが，本稿の検証手法を決定木集合モデル一般に適用する際には，学習過程に配慮する必要がある．このようなとき，はじめに冗長なパスの削除を行うことで決定木の構造を変え，偽反例を回避できる可能性もある．

文献 [15] の従来ツールと比較したときの，モデルの縮約にともなう検証時間の悪化の要因には，縮約のための計算による直接的なものと，論理式の構造の変化から起こる間接的なものがあげられる．直接的な要因は検証プログラム上で特定可能であり，処理時間の計測結果を 3.3 節に示した．サンプルとした XGBoost モデルでは，検証時間への影響が無視できる程度に縮約処理の時間は短かい．間接的な要因についてはプログラム上で特定することはできないが，近似閾値が 0 のときの検証時間を従来ツールと比較することで評価した．表 1 に示したように近似閾値が 0 のときは検証時間を約 25%増加させるが，近似閾値 20,000 で決定木を 1 つ近似した段階で，縮約の効果の方が上回る．

以上から，モデルの縮約は検証精度を確保しつつ検証時間を短縮するうえで有効であると結論できる．一方，サンプルとした小規模モデルでは，モデルの縮約を行わなくても形式検証が可能であり，モデルの縮約の効果が見えにくい．充足可能性探索の計算量が，最悪の場合は決定木の数 K の指数関数のオーダ $O(L^K)$ であるのに対し，縮約により h 個の決定木を近似した場合には $O(L^{K-h})$ であり，縮約に必要な計算量は線形関数のオーダ $O(L \times K)$ であることから，より規模の大きなモデルに対してはモデルの縮約による検証効率向上の効果は顕著になると考えられる．

5. 関連研究

本稿は，決定木をパスの論理積で表現することで XGBoost モデルを論理式に変換し，SMT ソルバを使って振舞いを検証する，Sato ら [14] の研究をもとにしている．このとき，決定木の数が増加するに従ってパスの組合せ数が指数関数的に増大するため，充足可能性探索が困難になることが課題であった．これに対し本稿では，1 つの決定木では必ず 1 つのパスが選ばれることを前提に，式 (2) のように決定木をパスの論理和で表現する．これにより，前提条

件を満たさないパスを充足可能性探索の前に削除して，計算量を削減することができる．さらに，推論出力データへの寄与が小さい決定木の式を定数値の範囲 (3) で置き換えることで，充足可能性探索を行う決定木の数を減少させる．

深層ニューラルネットワーク（DNN）の振舞いの形式検証の分野では，様々な研究がなされている．Huang ら [7] は，DNN を使った画像分類において元画像の近傍にある画像の分類不変性を検証する方法を提案し，SMT ソルバに基づいた自動検証フレームワークを提供した．Reluplex [9] は，DNN を論理式に変換するアルゴリズムの提案である．このアルゴリズムは，DNN の構成要素として広く使われる Rectified Linear Unit（ReLU）活性化関数に対応する．論理式への変換によって，DNN の振舞いが検証性質を満たすことを SMT ソルバを使って検証できる．DNN の規模が大きくなるにつれて充足可能性探索の計算量が指数関数的に増加することが課題であり，Elboher ら [4] はニューロンを合併することで DNN の規模を縮小する近似方法を提案した．この方法は健全であり，近似した DNN が検証性質を満たす場合にはもとの DNN も検証性質を満たすことが保証できる一方，反例が検出された場合には偽反例である可能性がある．ニューロンの合併は発見的方法で行うが，反例にガイドされた近似の洗練方法が示されている．

Sharma ら [16] は，機械学習によって作成された AI モデルの公平性テストにおいて，AI モデルの振舞いを模倣する決定木を機械学習により作成し，決定木について検出した反例から AI モデルをテストするデータを導出する方法を提案した．テスト対象の AI モデルをブラックボックステストするのに対し，その振舞いを学習データとして作成した決定木はホワイトボックスモデルと位置づけられ，決定木を論理式に変換し SMT ソルバを使って検証性質の充足可能性探索を行う．反例が得られた場合には反例をテストデータとして AI モデルをテストし，AI モデルも同様に検証性質に違反する場合には AI モデルの誤りと判定する一方，AI モデルは検証性質に違反しないときは決定木を再学習するための学習データとして利用する．決定木で反例が得られない場合でも AI モデルが検証性質を満たしているとはいえないので，検証性質が満たされないことが見込まれる場合の違反の検出に有効である半面，検証性質が満たされることの検証には適さない．

6. まとめ

本稿では，多数の決定木から構成される AI モデルの振舞いの形式検証について述べた．形式検証では前提条件と検証性質を与え，前提条件を満たす入力データすべてに対して推論出力データが検証性質を満たすことを論理的に検証する．これは AI モデルの推論の正しさの検証ではないが，推論出力データが想定した範囲内にあることを保証できる点で有用性が高い．一方，決定木の間のパスの組合せについて検証するので，最悪の場合の計算量は，1 つの決定木あたりのパス数を底とする決定木数の指数関数になる．この課題に対して本稿では，パスと決定木の数を減らして計算量を削減する，AI モデルの縮約方法を提案した．計算量が決定木数の指数関数であることに変わりはないが，パス数と決定木数を減らすことで，モデルの規模に対する計算量の急速な増加を抑制する．縮約により検証精度が下がると偽反例が検出される可能性があるが，近似閾値に基づいて縮約を行うことで，検証時間と検証精度が制御できる．

ツールを使ったケーススタディにより，モデルの縮約の効果を示した．前提条件，検証性質，近似閾値を与えるとツールは機械的に検証を行い，ケーススタディの AI モデルについては現実的な時間内に検証を終えた．検証により AI モデルが設計上想定していない振舞いをしないことが保証できるだけでなく，前提条件や検証性質を変えて検証を繰り返すことで AI モデルの振舞いを理解することができる．より規模の大きい AI モデルへの適用を通じて，検証時間と検証精度をバランスさせる手法を洗練し，実用化研究を進めることが今後の課題である．

参考文献

[1] Chen, T. and Guestrin, T.: XGBoost: A Scalable Tree Boosting System, *Proc. 22nd ACM SIGKDD*, pp.785–794 (2016).

[2] Czarnecki, K. and Salay, R.: Towards a Framework to Manage Perceptual Uncertainty for Safe Automated Driving, *Proc. WAISE 2018* (2018).

[3] Ehlers, R.: Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks, arXiv: 1705.01320v3 [cs.LO] (2017).

[4] Elboher, Y.Y., Gottschlich, J. and Katz, G: An Abstraction-Based Framework for Neural Network Verification, *Proc. CAV 2020*, LNCS 12224, pp.43–65, Springer (2020).

[5] Garlan, D.: Software Engineering in an Uncertain World, *Proc. FoSER 2010*, pp.125–128 (2010).

[6] 林　晋：プログラム検証論，共立出版 (1995).

[7] Huang, X., Kwiatkowska, M., Wang, S. and Wu, M.: Safety Verification of Deep Neural Networks, *Proc. CAV 2017*, LNCS 10426, pp.3–29, Springer (2017).

[8] 石川冬樹，來間啓伸，中島　震：不確かさを考慮したソフトウェア・テスティングおよび形式検証，情報処理，Vol.58, No.8, pp.693–695 (2017).

[9] Katz, G., Barrett, C., Dill, D., Julian, K. and Kochenderfer, M.: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, *Proc. CAV 2017*, LNCS 10426, pp.97–117, Springer (2017).

[10] 來間啓伸，佐藤直人，中川雄一郎，小川秀人：演繹的開発手法と帰納的開発手法の結合に基づく機械学習適用ソフトウェアの形式検証とテスト，情報処理学会論文誌，Vol.61, No.2, pp.407–416 (2020).

[11] 來間啓伸，明神智之，佐藤直人，小川秀人：XGBoost 法により学習させた AI モデルのふるまいの形式検証，ソフトウェアエンジニアリングシンポジウム 2021 論文集，pp.137–142 (2021).

[12] 丸山　宏：機械学習工学に向けて，日本ソフトウェア科学会第 34 会大会講演論文集 (2017).

[13] 中島 震：ソフトウェア工学から学ぶ機械学習の品質問題，丸善出版 (2020).
[14] Sato, N., Kuruma, H., Nakagawa, Y. and Ogawa, H.: Formal Verification of a Decision-Tree Ensemble Model and Detection of Its Violation Ranges, *IEICE Trans. & Syst.*, Vol.E103-D, No.2, pp.363–378 (2020).
[15] 佐藤直人，小川秀人，來間啓伸，明神智之：AI ソフトウェアのテスト，リックテレコム (2021).
[16] Sharma, A. and Wehrheim, H.: Automatic Fairness Testing of Machine Learning Models, *Testing Software and Systems*, *ICTSS 2020*, Casola, V., et al. (Eds.), LNCS 12543, pp.255–271 (2020).
[17] 土屋達弘：充足可能性判定を利用したモデル検査，コンピュータソフトウェア，Vol.29, No.1, pp.19–29 (2012).
[18] 梅村晃広：SAT ソルバ・SMT ソルバの技術と応用，コンピュータソフトウェア，Vol.27, No.3, pp.24–35 (2010).
[19] House Sales in King Country, USA, available from ⟨https://www.kaggle.com/harlfoxem/housesalesprediction⟩

來間 啓伸 （正会員）

1981 年広島大学理学部卒業．1983 年同大学大学院理学研究科博士課程前期修了．1984 年同博士課程後期中退，株式会社日立製作所入社．主としてソフトウェア工学と形式手法の研究に従事．2006 年総合研究大学院大学複合科学研究科情報学専攻博士課程修了．博士（学術）．日本ソフトウェア科学会，IEEE，ACM 各会員．

明神 智之

2006 年電気通信大学大学院情報システム学研究科博士前期課程修了．同年株式会社日立製作所入社．組込みシステム，ソフトウェアテスティング，形式手法，機械学習工学等の研究開発に従事．

佐藤 直人 （正会員）

2003 年東京工業大学情報工学科卒業．2005 年同大学大学院情報理工学研究科計算工学専攻修士課程修了．同年株式会社日立製作所入社．ソフトウェアテスト技術，形式手法，モデルベース開発技術，量子ソフトウェア開発技術などの研究開発に従事．博士（工学）．

小川 秀人 （正会員）

1996 年名古屋大学大学院工学研究科博士前期課程修了．同年株式会社日立製作所入社．ソフトウェアテスティング，形式手法，機械学習工学等の研究開発に従事．2015 年北陸先端科学技術大学院大学博士後期課程修了．博士（情報科学）．電子情報通信学会，日本ソフトウェア科学会各会員．