XGBoost 法により学習させた AI モデルのふるまいの形式検証

來間啓伸[1]　明神智之[1]　佐藤直人[1]　小川秀人[1]

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 モデルのふるまいを予測することは困難であり，仕様の充足性を確認することを目的とした従来のテスト手法は適用できない．AI モデルのふるまいの形式検証は，指定した入力データ範囲の中で AI モデルの出力データが検証性質を満たすことを網羅的に検証し，不適切なデータが出力されないことを確認できる点で，この課題の解決に有効である[4]．その反面，一般に形式検証には多くの計算が必要であり，検証が現実的な時間内に終了しない場合がある．本稿では XGBoost 法を使って学習させた決定木集合モデルの形式検証について述べるとともに，検証時間と検証精度を制御する手段として近似閾値を用いたモデルの縮約を提案する．米国の住宅価格データベース[7]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，形式検証のフィージビリティを評価した．検証ノウハウの蓄積とツールの洗練が，今後の課題である．

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

Formal Verification of Properties of AI Models Trained by XGBoost Machine-Learning Method

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

Abstract: Machine learning is expected to be a development method of software which process phenomena that are difficult to analyze and describe logically. Since machine learning develops AI models inductively, traditional deductive techniques are not applicable for testing them. Consequently, the behavior of AI models may be inconsistent with the software design. Formal verification of AI models solves this problem by checking that the output data of the model satisfies the given property for every input data specified in the precondition [4]. However, much computing resources are needed for formal verification and frequently verification does not terminate in a practical time. In this paper, we describe a formal verification method for decision tree ensemble models trained using XGBoost machine learning method and propose a model reduction technique which uses approximation threshold to control 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 [7]. Our feature work is to extend the case study to larger models and refine our verification tool.

Keywords: Machine Learning, Decision Tree Ensemble, Testing of AI Software, 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ついて動作する Cyber 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[2]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処理を担ってきた．しかし，CPS では実世界との結びつきが密になるので，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．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推論のための特徴を学習し，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．

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であるため，満たすべき仕様が存在しない．このため，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[3]．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運用時に AI モデルに入力され得るデータのサンプルにすぎないので，データ数の点でもカバレッジの点でも十分なテストとは言い難い．注意深く学習が進め

1 株式会社日立製作所　研究開発グループ
Research and Development Group, Hitachi, Ltd.

られた AI モデルであっても，学習データセットにない入力データに対しては，不適切であることが明らかな推論結果を出力する可能性がある．

システム開発で AI モデルを利用する観点からは，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であれば妥当な推論結果が期待できるのか，を把握することが重要になる．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 モデルを分析して原因を解明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ソフトウェアを設計することもできる．

本稿では，機械学習法の一つである XGBoost[1]を使って学習させた AI モデルを対象とした，AI モデルのふるまいの形式検証について述べる．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，範囲内の全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，XGBoost を使って学習させた AI モデルを XGBoost モデル，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，システム内の他のコンポーネントとのインターフェイスから要請される部分仕様から設定する．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（counter example）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，人が指定した期待とは異なる点で，精査すべき具体例である．

以下，２節では 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節では，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節で，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るモデルの例を図 2.1 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，入力データにしたが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 2.1 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全ての決定木について足し合わせた値である．図 2.1 のモデルでは，例えば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．

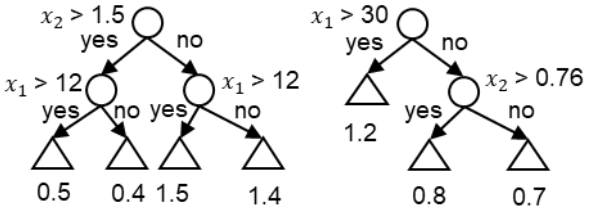

図 2.1 XGBoost モデルの例
Fig. 2.1 An example of XGBoost model

2.2 形式検証

XGBoost モデルの形式検証では，入力データがx，推論出力データがyのとき，前提条件$P(x)$と検証性質$I(y)$に対して XGBoost モデルが

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

を満足することを検証する[4]．具体的には，XGBoost 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積で結合した

$$R(x,y) \wedge P(x) \wedge \neg I(y)$$

が真になるxの値を，充足可能性問題として SMT（Satisfiability Modulo Theories）ソルバ[6]を使って機械的に

探索する．そのようなxの値が見つかれば，その値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，全体が真になるxの値が見つからなければ，全ての入力データについて，検証性質が満たされるかあるいは前提条件外であり，いずれの場合でも前提条件の範囲内では検証性質が満たされることが言える．

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とすると，入力データxがノードmに与えられた時エッジ$< m,n >$が選ばれる条件$e_{m,n}$は，

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

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，入力データxに対してルートノード$n_0^{(k)}$からリーフ$l_i^{(k)}$に到達するパスが選ばれる条件$q_i^{(k)}(x)$は，

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

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，

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

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

である．ここで、$\bigwedge_{j=0}^{m}$ は∧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，リーフに到達する全てのパスについての下記の論理和

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

である．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

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

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，例えば図 2.1 の XGBoost モデルは，次の論理式に変換することができる．

$$\begin{aligned}
&((x_2 > 1.5) \wedge (x_1 > 12) \wedge \left(y^{(1)} = 0.5\right) \vee \\
&\ (x_2 > 1.5) \wedge (x_1 \le 12) \wedge \left(y^{(1)} = 0.4\right) \vee \\
&\ (x_2 \le 1.5) \wedge (x_1 > 12) \wedge \left(y^{(1)} = 1.5\right) \vee \\
&\ (x_2 \le 1.5) \wedge (x_1 \le 12) \wedge \left(y^{(1)} = 1.4\right)) \wedge \\
&((x_1 > 30) \wedge \left(y^{(2)} = 1.2\right) \vee \\
&\ (x_1 \le 30) \wedge (x_2 > 0.76) \wedge \left(y^{(2)} = 0.8\right) \vee \\
&\ (x_1 \le 30) \wedge (x_2 \le 0.76) \wedge \left(y^{(2)} = 0.7\right)) \wedge \\
&(y = y^{(1)} + y^{(2)})
\end{aligned}$$

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，指定した入力データの範囲について検証性質が満たされる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，推論出力データが正しいことを個々に与えたテストデータについて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は，

$$R(x,y) \wedge P(x) \wedge \neg I(y)$$

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

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

$$\left(y = \sum_{k=1}^{K} y^{(k)}\right) \wedge P(x) \wedge \neg I(y)$$

と変形することができ，最悪の場合には全ての決定木k（$1 \le k \le K$）についてパスi（$1 \le i \le L^{(k)}$）の組み合わ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，パスの組み合わせの総数は

$$L^K$$

となることから，最悪の場合の計算量は決定木の数を指数とする指数関数になる．

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

$$R(x,y) \wedge P(x) \wedge \neg I(y)$$

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

$$(y = \sum_{k=1}^{K} y^{(k)}) \wedge \neg I(y)$$

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

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

を予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削除

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

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

を

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

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た可能性がある．本稿では，真の値が検証性質を満たすにも関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．反例が検出された時，真の反例か偽反例かの区別は，近似前のモデルに反例を入力して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得る．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. 抽出した決定木が複数の場合，最大値の和と最小値の和を計算
3. 残りの決定木について充足可能性探索式を作り，推論出力データyの値を決定木の出力$y^{(k)}$の和と最大値の和，最小値の和で置き換え

$$y \leq \left(\Sigma\, y^{(k)} + \text{最大値の和}\right) \wedge$$
$$y \geq (\Sigma\, y^{(k)} + \text{最小値の和})$$

近似対象とする決定木の抽出に要する計算量のオーダーは，$O(L \times K)$である．これに対し，近似する決定木の数をhとすると，充足可能性探索の計算量のオーダーは最悪の場合でも$O(L^{K-h})$となる．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 2.2 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$y^{(2)}$として決定木の論理式を

$$0.7 \leq y^{(2)} \wedge y^{(2)} \leq 1.2$$

で置き換える．図 2.2 の縮約したモデルを変換すると，次の論理式が得られる．

$$((x_2 > 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 0.5) \vee$$
$$(x_2 \leq 1.5) \wedge (x_1 > 12) \wedge (y^{(1)} = 1.5)) \wedge$$
$$(0.7 \leq y^{(2)} \wedge y^{(2)} \leq 1.2) \wedge$$
$$(y = y^{(1)} + y^{(2)})$$

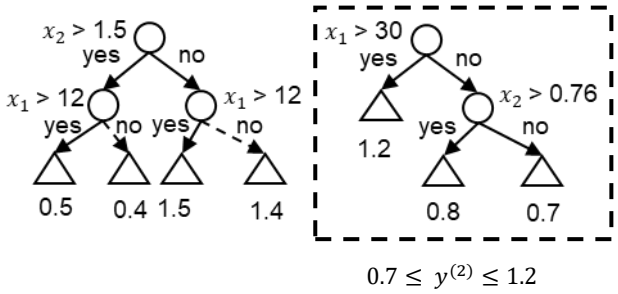

図 2.2 モデルの縮約

Fig. 2.2 Model reduction

3. ケーススタディ

3.1 検証対象モデル

モデルの縮約の効果を評価するため，米国の住宅価格データベース[7]を学習データとして XGBoost を使って学習させた住宅価格予想モデル[5]をサンプルに，検証ツールを使った検証時間を測定した．このモデルは，

- リビングルームの面積(290 から 13540 平方フィート)
- 寝室の数(0 から 33)
- 土地面積(520 から 1651359 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 9410 平方フィート)
- 地下室の面積(0 から 4820 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 72 個，7 の決定木 24 個，6 の決定木 4 個の合計 100 個の決定木からなり，パスの組み合わせの総数は約$8^{100} \cong 2 \times 10^{90}$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：全ての住宅
  検証性質：予想価格は 10,000,000 ドル未満
- 検証内容 2
  前提条件：全ての住宅
  検証性質：予想価格は 100,000 ドル以上
- 検証性質 3
  前提条件：リビングルームの面積が 7000 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：予想価格は 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

形式検証は，2.3 節のモデルの縮約を行わない場合と，近似閾値を変えてモデルを縮約した場合について行った．検証時間の比較を，表 3.1 から表 3.3 に示す。検証に用いた機器の CPU は Intel Core i7-10700K 3.8GHz，メモリは 32GB，OS は Windows 10 Pro 64bit である2.

表 3.1 は，検証内容 1 の検証結果である．検証内容 1 については反例が存在せず，サンプルとしたモデルでは全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．前提条件が「全ての住宅」であるため冗長なパスは存在せず，モデルの縮約では決定木の近似のみを行っている．ただし，ツールは冗長なパスが存在するかどうかを調べており，その処理時間が検証時間に含まれる．表中の近似決定木は，ウェイトの最大値と最小値の差が近似閾値以下の，近似対象となる決定木の数である．最大誤差は，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である．

近似閾値 80000 までの全ての場合について，反例は検出されなかった．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 80000 のときも十分な検証精度があると言える．一方，検証時間は，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短くなる傾向がみてとれる．

表 3.1　検証内容 1 の検証時間
Table 3.1 Verification time for property 1

近似閾値	近似決定木	最大誤差	検証時間 (sec)
縮約なし	-	-	1381.33
0	0	0	1138.39
20000	1	17344	978.18
40000	3	93814	1006.56
60000	9	387504	770.72
80000	16	891273	536.14

検証内容 2 については，予測価格が 100,000 ドル未満になり得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，反例が存在しない場合に比べて表 3.2 に示すように検証時間は短い．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．

検証内容 1 と同様に検証内容 2 も前提条件が「全ての住宅」であるため冗長なパスは存在せず，モデルの縮約では決定木の近似のみを行うが，冗長なパス検出の処理時間が検証時間に含まれる．近似による最大誤差は検証性質の 10,000 ドルに比べると大きく影響が無視できないと考えられるが，近似閾値 40000 までは正しく反例が検出された．一方，近似閾値 60000 と 80000 では偽反例が検出されている．前述のように，充足可能性探索により具体的な入力データが得られるので，得られた入力データを近似前のモデルに入力して推論出力データが検証性質を満たすか否かを調べることで，反例か偽反例かの判定は容易に行うことが

2 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.

できる．偽反例が検出された場合には検証結果が確定しないので，近似閾値を下げて再度検証を行う必要がある．

表 3.2　検証内容 2 の検証時間

Table 3.2 Verification time for property 2

近似閾値	近似決定木	最大誤差	検証時間 (sec)
縮約なし	-	-	0.38（反例）
0	0	0	1.04（反例）
20000	1	17344	0.40（反例）
40000	3	93814	0.17（反例）
60000	9	387504	0.15（偽反例）
80000	16	891273	0.35（偽反例）

検証内容 3 の形式検証では，表 3.3 に示すように近似閾値が 80000 の場合を除いて反例が検出されず，60000 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言える．一方，近似閾値が 80000 の時は検証精度が不十分であり，偽反例が検出されている．

検証内容 3 には有効な前提条件があるため，前提条件を満たさない冗長なパスがモデルに存在し，モデルの縮約では冗長なパスの削除と決定木の近似を行っている．決定木の近似は冗長なパスを削除した後に行うので，同じ近似閾値に対しても検証内容 1，2 より近似される決定木の数が多い．近似される決定木の数が増えるにしたがって検証時間は短くなる傾向にあるが，縮約しない場合に比べて冗長なパスを削除する処理が加わる分は検証時間が長くなる．冗長なパスを削除する処理時間は決定木の数に比例し，パス削除による検証時間の削減効果は決定木の数の指数関数に比例することから，検証対象の XGBoost モデルの決定木数が多い場合には効果がより明確になると考えられる．

表 3.3　検証内容 3 の検証時間

Table 3.3 Verification time for property 3

近似閾値	近似決定木	最大誤差	検証時間 (sec)
縮約なし	-	-	42.65
0	3	0	51.72
20000	5	30469	43.78
40000	12	261343	47.79
60000	20	648751	30.05
80000	29	1307918	0.91（偽反例）

以上より，2.3 節のモデルの縮約は，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なわなくても形式検証が可能であり，縮約処理に要する時間を考慮すると，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の指数関数になることから，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．

4. まとめ

本稿では，多数の決定木から構成される AI モデルのふるま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全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み合わせについて検証するので，最悪の場合の計算量は決定木の数の指数関数になる．この課題に対して本稿では，AI モデルを縮約することで計算量を削減する方法を提案した．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ふるま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 モデルのふるまいを理解することができる．より大規模な AI モデルを対象に，検証時間と検証精度をバランスさせる検証ノウハウの蓄積とツールの洗練が，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. the 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] 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.
[4] Sato, N., Kuruma, H., Nakagawa, Y., Ogawa, H.: Formal Verification of a Decision-Tree Ensemble Model and Detection of Its Violation Ranges, IEICE Trans. & Syst. E103-D (2), pp.363 – 378, 2020.
[5] 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.
[6] 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.
[7] House Sales in King Country, USA https://www.kaggle.com/harlfoxem/housesalesprediction