

学習モデルグラフ上での仮説検証に基づく機械学習モデル生成方法の 提案と自動車センサデータへの適用評価

白崎 悠太^{†1*} 永井 利幸^{†1} 青山 幹雄^{†2}

概要: 機械学習を用いたシステム開発の必要性が高まっている。要求認識精度を達成する機械学習モデル生成にはフィーチャ(特徴量)設計の利用が提案されている。しかし、学習へのフィーチャの影響分析は未確立である。本稿では学習モデルグラフ上での仮説検証に基づく機械学習モデル生成方法を提案する。プロパティグラフモデルによって学習過程を学習モデルグラフとしてモデル化する。学習過程における重みパラメータの変化率の分析によってプロパティグラフ上で学習に影響を与えるフィーチャノードを特定し入力フィーチャを制御することで機械学習モデルを生成する。プロトタイプを実装し自動車センサデータに適用することにより提案方法の有効性と妥当性を評価する。

キーワード: 機械学習ソフトウェア工学, フィーチャ設計, フィーチャ選択, プロパティグラフ, グラフデータベース

A Generation Method of Machine Learning Models Based on Hypothesis Testing on Learning Model Graphs and Its Evaluation with Automotive Sensor Data

YUTA SHIRASAKI^{†1*} TOSHIYUKI NAGAI^{†1} MIKIO AOYAMA^{†2}

1. はじめに

機械学習の発展とともにそれを応用して解決可能な問題領域が広がっている。様々なドメインにおいて機械学習を用いたソフトウェアシステムを開発する必要性が高まっており、多くの場合、教師あり学習によって機械学習モデルを生成している。しかし、機械学習アルゴリズムを用いるだけでは要求する認識精度を達成するようなモデルを生成することは困難である。要求される認識精度を達成するために、学習モデルの最終的な認識精度に基づく仮説検証によってフィーチャ(特徴量)設計とモデルのチューニングが繰り返行われている。しかし、学習へのフィーチャの影響分析が未確立であるため、フィーチャの変更によって学習が効率的に行われているかを分析できない。

本稿ではフィーチャ設計を活かした学習モデル生成プロセスの確立を目的として、学習過程をプロパティグラフモデルでモデル化することによって、プロパティグラフ上での仮説検証に基づく機械学習モデル生成方法を提案する。

2. 研究課題

学習過程の可視化を目的としたプロパティグラフの利用、フィーチャの影響分析、実データへの適用は、機械学習を用いたシステム開発において実践されていない。本稿では、研究背景を踏まえ以下の3点を研究課題として設定する。

- (1) プロパティグラフによる機械学習モデルのモデル化方法の確立

- (2) プロパティグラフによるフィーチャ分析方法の確立
- (3) 実データへの適用による有効性と妥当性の評価

3. 関連研究

3.1 機械学習

機械学習はデータから自動的にパターン認識し予測を行う技術である[4]。データから自動的に学習しモデルを生成するため機械学習の学習過程はブラックボックスであることが知られている。学習を効率的に行うために、どのフィーチャからモデルを生成するかフィーチャ選択方法が提案されている[9]。また、フィーチャを制御し学習を効率化するために画像データに対してフィーチャの学習への貢献を分析する方法が提案されている[8]。しかし、認識精度からの分析であり学習へのフィーチャの影響を十分に分析できていない。

深層学習の利用を容易にするために、深層学習フレームワークが提供されている。主な深層学習フレームワークとして、Preferred NetworksのChainer[16]やGoogleのTensorFlow[5]がある。特に、Chainerはデータを処理(Run)しながら、計算グラフを構築(Define)するDefine-by-Run方式を採用することにより、学習対象によって柔軟にモデルを変更可能という特徴がある。

3.2 機械学習モデルの生成方法

機械学習モデルの生成方法にはデータのみから生成する方法、表現学習しフィーチャを抽出する方法、転移学習がある[4]。

3.3 フィーチャ工学 (Feature Engineering)

フィーチャ工学とは、機械学習の適用対象となる問題の本質

^{†1} 南山大学大学院 理工学研究科 ソフトウェア工学専攻
Graduate Program of Software Engineering, Nanzan University

^{†2} 南山大学 理工学部 ソフトウェア工学科
Dep. of Software Engineering, Nanzan University

* 現在、株式会社デンソー勤務

を表現したフィーチャをデータから生成する技術体系である[14]. 機械学習の予測精度と学習効率向上を期待できる. フィーチャ設計とは, 主に探索的データ分析, データクレンジング, フィーチャ生成, 変換, 選択などのアクティビティから構成され, モデリングの結果によって検証され反復的に行われる. フィーチャ設計の応用として, 様々なデバイスから収集される IoT データを分析するための反復的フィーチャ設計を自動化する方法が提案されている[1]. また, 強化学習を用いてフィーチャ設計を自動化する方法が提案されている[7]. データ構造によってフィーチャ設計プロセスは異なり, 画像, 文章, グラフなどに対しては深層学習を用いてフィーチャを抽出する表現学習が提案されている[3].

3.4 グラフモデル

3.4.1 プロパティグラフ

プロパティグラフとはデータ間の意味定義を表現可能とするセマンティックグラフの拡張である. ノードとエッジに属性の集合をプロパティとして付与可能である[12, 13]. 図 1 にプロパティグラフの例を示す.

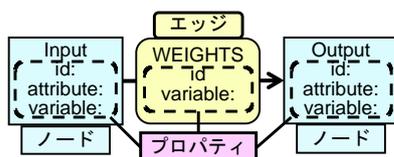


図 1 プロパティグラフ
 Figure 1 Property Graph

3.4.2 プロパティグラフに基づく分析

グラフモデルはソーシャルネットワークをはじめとした様々なコミュニティ構造の分析に応用されている[2]. 発話データをプロパティグラフでモデル化し, ステークホルダ間の関係を分析することによってステークホルダの役割を明らかにする方法が提案されている[17]. また, ERP(Enterprise Resource Planning)のためのデータをプロパティグラフ上で分析しデータマイニングする方法が提案されている[15].

3.5 グラフデータベース

プロパティグラフを扱うデータベースとして, グラフデータベースがある[6]. スキーマレスなデータベースであるので, グラフ構造の変化に対応した分析が可能である. 実装として Neo4j[11], JanusGraph[19]などがある.

4. アプローチ

4.1 フィーチャ工学の導入による学習

本稿では, 教師あり学習を対象とする. 教師あり学習によるモデル生成では, 学習モデルがとることのできる入力フィーチャの構造や学習アルゴリズムに適した方法でデータを抽象化しフィーチャを獲得するフィーチャ設計を行う必要がある. 特に, フィーチャをテーブルデータから抽象化する場合にはドメイン知識に基づいてフィーチャ選択を反復的に行う方法が効果的である.

しかし, データ駆動のアプローチではモデルの予測結果から修正点を特定することは容易ではなく, 試行錯誤を繰り返すことによるコストがかかってしまう. そのため, フィーチャ設計がモデル生成におけるプロセスの中で最も時間的コストがかかるプロセスとなっている[7]. この点に着目し, 本稿では, 学習に対して支配的なフィーチャを特定することによって学習を制御する機械学習モデル生成方法を提案する.

提案方法では, ニューラルネットワークの学習過程からフィーチャ分析を行うアプローチをとる. 学習過程をモデル化し, フィーチャ分析を可能とするため, 新たに, 学習モデルグラフを提案する. 学習モデルグラフとして, 学習モデルの構造的観点からグラフによってモデル化可能, かつバッチサイズやエポックなどの学習に関するデータを付加可能なプロパティグラフを採用した.

4.2 仮説検証型反復学習プロセス

データに対するプロパティグラフ上での仮説検証型の反復学習プロセスを提案する(図 2). ドメイン知識からモデルの入力とするフィーチャを選択する際, どのフィーチャが認識精度向上に役立つかは仮説でしかない. したがって, 実際にモデルを生成しなければ分からない. 従来, 試行錯誤であった特徴量選択の仮説検証を効率的に行うために学習に対するフィーチャの影響に着目し分析する. 提案方法では, 反復を通して各プロセスを詳細化する. 学習モデルの構造をプロパティグラフとして表現する. 学習モデルの学習過程における最終的な認識精度には現れないパラメータの変化に着目する. プロパティグラフ上で学習過程における学習モデルの差分を分析することによって学習に影響を与えるフィーチャを特定し, それに基づいて学習する.

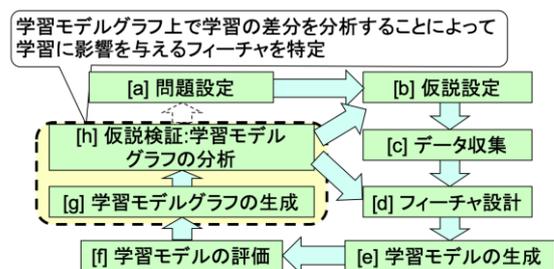


図 2 反復学習プロセス

Figure 2 Iterative Learning Process

5. 学習モデルグラフ上での仮説検証型機械学習モデル生成方法

5.1 学習モデルグラフの定義

学習モデルグラフとは, 学習モデル構造と学習のための付加データをプロパティグラフとして表現したグラフである.

(1) 学習モデルグラフのメタモデル

学習モデルグラフ生成のためにニューラルネットワークの学習モデル構造を表現可能なプロパティグラフモデルを定義する. 学習によって得た学習モデルデータから生成する学習モデル

グラフのメタモデルの定義を図3に示す。FeatureからOutputまでが学習モデル構造を表現しており、Objective Function ノードとOptimizer ノードで学習に必要な設定データを表現している。1-LAYER WEIGHTS から n-LAYER WEGHTS まではニューラルネットワークの線形変換と活性化関数を含む層を繰り返す構造を表現している。表1にノードの定義と付与するプロパティ、表2にエッジの定義と付与するプロパティ、表3にノードとエッジに付与したプロパティの定義を示す。

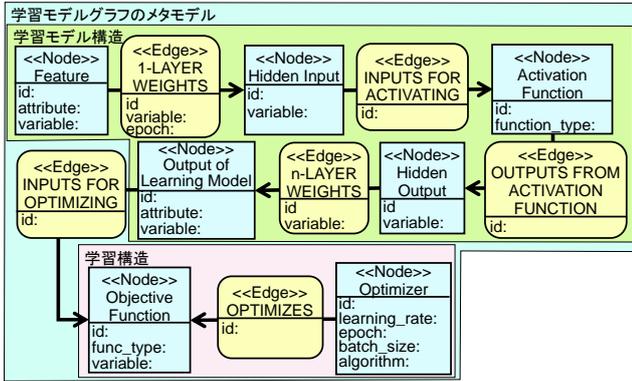


図3 学習モデルグラフのメタモデル

Figure 3 Metamodel of Learning Graph Model

表1 ノードの定義

Table 1 Definition of Nodes

ノード	定義	プロパティ
Feature	学習モデルへの入力 (フィーチャ)	attribute, variable
Hidden Input	活性化関数への入力	variable
Activation Function	活性化関数	id, func_type
Hidden Output	活性化関数の出力	variable
Output of Learning Model	モデルの最終出力	attribute, variable
Objective Function	目的関数	id, func_type, variable
Optimizer	最適化情報を保持するノード	id, learning_rate, epoch, batch_size, algorithm

表2 エッジの定義

Table 2 Definition of Edges

エッジ	定義	プロパティ
1-LAYER WEIGHTS	各ノード間をつなぐ第1層の重み付け	id, variable, epoch
INPUTS FOR ACTIVATING	活性化関数ノードへの入力	id
OUTPUTS FROM ACTIVATION FUNCTION	活性化関数ノードからの出力	id
n-LAYER WEIGHTS	各ノード間をつなぐ第n層の重み付け	id, variable
INPUTS FOR OPTIMIZING	目的関数への入力	id
OPTIMIZES	パラメータを最適化	id

表3 プロパティの定義

Table 3 Definition of Properties

プロパティ	定義
id	ノードとエッジの id
attribute	フィーチャの属性名
variable	ノードとエッジが保持する値
func_type	関数名
learning_rate	学習率
epoch	学習数
batch_size	バッチサイズ
algorithm	最適化アルゴリズム

5.2 提案プロセス

図2のプロセスを詳細化した提案プロセスを図4に示す。

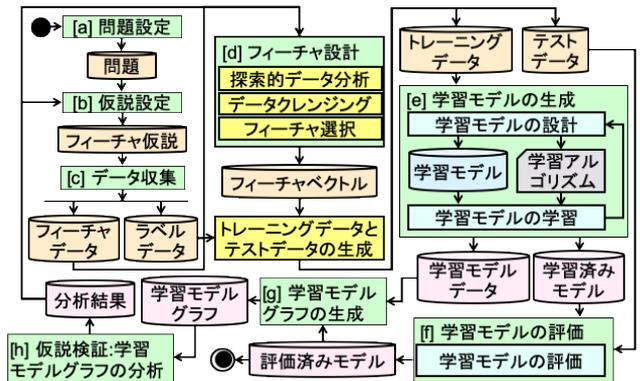


図4 提案プロセス

Figure 4 Proposed Process

[a] 問題設定

機械学習モデルが要求を満たすためにモデルが解くべき問題を設定する。プロセスを繰り返すにつれて問題設定は詳細化される。

[b] 仮説設定

問題解決に必要なだと推定されるデータについての仮説を設定する。学習モデルグラフの分析結果から分析対象として収集すべきデータを見直す。

[c] データ収集

仮説から分析に必要なデータを取得する。

[d] フィーチャ設計

学習モデルの入力となるフィーチャベクトルを生成する。探索的データ分析、データクレンジング、フィーチャ選択の3ステップからなる。

[e] 学習モデルの生成

トレーニングデータを用いて学習モデルを設計と学習の2ステップから生成する。

[f] 学習モデルの評価

テストデータを用いて目的関数に基づき学習モデルの汎化誤差を評価する。モデルの評価結果が要求を満たしている場合、提案プロセスを終了する。

[g] 学習モデルグラフの生成

学習モデルデータからプロパティグラフとして差分を分析す

る学習モデルグラフを生成する。

[h] 仮説検証: 学習モデルグラフの分析

生成した学習モデルグラフから重みの変化率を分析することで、フィーチャの影響力を分析する。分析結果から仮説設定とフィーチャ設計を見直す。

5.3 フィーチャ設計

フィーチャデータを抽象化することによって学習モデルの入力となるフィーチャベクトルを生成するプロセスである。

(1) 探索的データ分析

フィーチャを抽出するための仮説を設定するために探索的データ分析を行う。探索データ分析では、データ間の分布と構造を可視化し分析することによって、有効だと推定されるデータクレンジング方法とフィーチャ選択の仮説を設定する。仮説はプロセスを繰り返すごとに詳細化する。

(2) データクレンジング

探索的データ分析で明らかになった仮説に基づきデータクレンジングを行う。データクレンジングでは、オリジナルデータからより特徴を表しているデータを生成するために欠損値の補完、不要なデータの削除、外れ値の削除をし、形式が異なるファイルから得られるデータを 1 つのデータ形式として再構成する。データクレンジングによってモデルの認識精度の向上が期待できる。

(3) フィーチャ選択

フィーチャ選択では、仮説に基づきデータから予測結果に影響を及ぼすと想定されるフィーチャを選択する。選択したフィーチャを標準化しフィーチャベクトルとして再構成する。標準化することによってそれぞれの特徴量の単位による分布の偏りを解消し予測精度の向上と学習の効率化が期待できる。

5.4 トレーニングデータとテストデータの生成

トレーニングデータとテストデータの生成の詳細について説明する。モデルの学習と評価に用いるトレーニングデータとテストデータを生成する。トレーニングデータ D_{train} とテストデータ D_{test} は、フィーチャベクトル F とラベル P のタプルであり、式(1)、式(2)で定義する。

$$D_{train} = (F_{train}, P_{train}) \quad (1)$$

$$D_{test} = (F_{test}, P_{test}) \quad (2)$$

トレーニングデータは、モデルの学習と訓練誤差の評価に用いる。テストデータはモデルの汎化誤差の評価に用いる。汎化誤差によって汎化能力を推定することが可能である。汎化能力とは学習していない未知のデータに対して学習済みモデルが高精度の予測ができるかの能力である。

5.5 学習モデルの生成

(1) 学習モデルの設計

モデル設計では、モデリングに必要なモデル構造と学習アルゴリズムを設計する。得られるフィーチャの形式と求められる出力の形式に基づいてモデルを設計する。モデル設計後に最適化アルゴリズム、学習数(epoch)、バッチサイズ、学習率を決定する。予測結果の精度によって学習モデルの設計を見直す。学習モデルの学習と評価に基づいて繰り返し行い、最適なモデル設計を確定する。

(2) 学習モデルの学習

トレーニングデータを用いて最適化アルゴリズムに基づいて学習モデルのパラメータを最適化する。学習の訓練誤差と汎化誤差の推移を観察し過学習を起こさないエポック数で学習を打ち切る。過学習とはトレーニングデータに対して過剰に適合し過ぎてしまった状態である。過学習している学習モデルはテストデータ(学習に使用しないデータ)に対して認識精度が上がりず実用に適さない。

5.6 モデルの評価

テストデータと評価関数を用いて生成した学習モデルの汎化性能を評価する。しかし、汎化能力に基づいてモデルのチューニングを行うため問題によってはテストデータとは別のデータセットを用意してハイパーパラメータチューニングの影響を受けないモデルの性能を評価する場合もある。さらに、データやパターンが多い場合にはテストデータを評価毎に選び評価し直す交差検証を実施することが重要である。

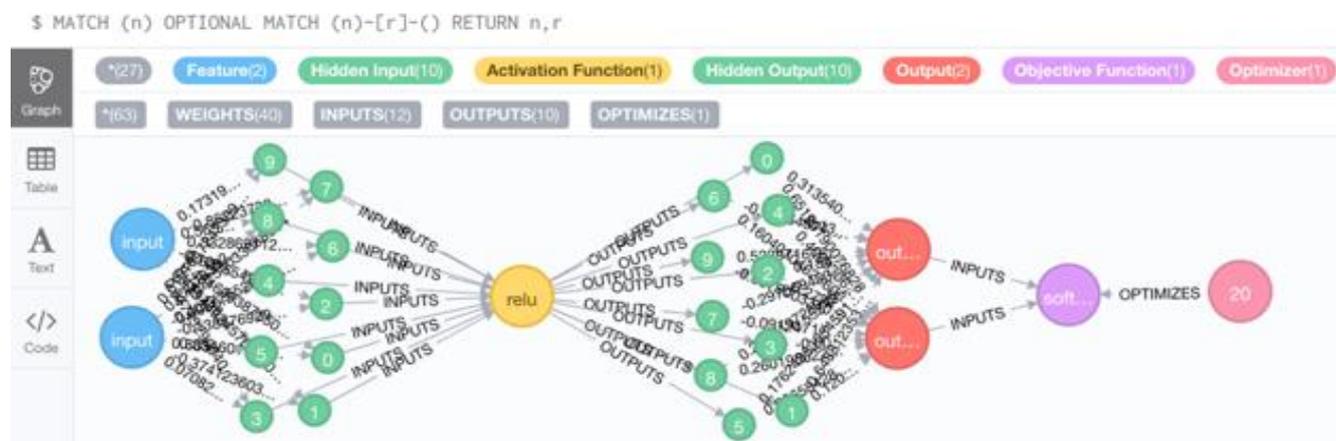


図5 学習モデルグラフの例
 Figure 5 An Example of Learning Model Graph

5.7 学習モデルグラフの生成

メタモデルの定義にしたがってグラフデータベース Neo4j にデータを挿入し学習モデルグラフを生成する。学習モデルグラフを生成し可視化した例を図 5 に示す。

5.8 学習モデルグラフの分析

学習過程で変化する重みパラメータを分析することで学習に対して影響力の強いフィーチャを特定する。重みの平均変化率の大きい重みと結ばれているフィーチャノードを特定することによって学習に影響を与えるフィーチャを特定する(図 6)。そのため学習数 e から学習数 e' における第 1 層 i 番目のフィーチャノードに連結する重みの平均変化率を式(3)により定義する。

$$R_j^{(l)} = \frac{1}{n} \sum_{i=1}^n \frac{|w_{i,j,e'}^{(l)} - w_{i,j,e}^{(l)}|}{|w_{i,j,e}^{(l)}|} \quad (3)$$

入力フィーチャ数 2, 出力数 2 である 1 層のニューラルネットワークを例にとって説明する。式(3)を第 1 層学習数 0 と学習数 1 の x_1, x_2 それぞれに適用し重みの平均変化率を求める。 x_1 と x_2 の重みの平均変化率を比較し大きい方を影響力が強いフィーチャとして特定する。学習数を重ねる場合も同様に学習後と学習前の重みの平均変化率を評価する。評価した重みの平均変化率の推移を分析することで学習過程における影響力が強いフィーチャを特定する。

2 層以上のニューラルネットワークにおいても同様に重みの平均変化率を求めるが 2 層以降のノード間は重みの平均変化率から影響力が強いフィーチャを特定することが困難である。本研究では、最もグローバルな特性を持つフィーチャを特定するため、浅い層を焦点とする。そのため、第 1 層におけるフィーチャ毎の重みの平均変化率を分析する。

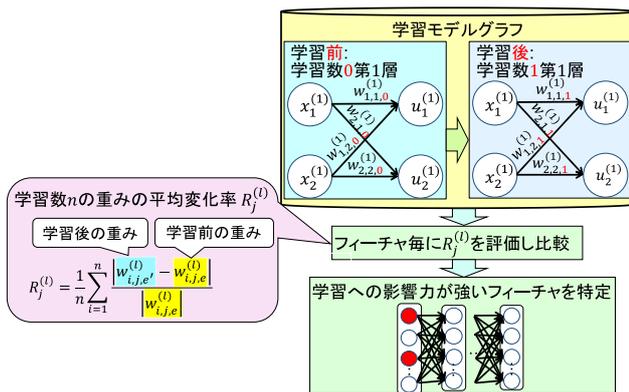


図 6 フィーチャの影響分析
 Figure 6 Influence Analysis of Features

6. プロトタイプの実装

6.1 実装環境

プロトタイプの実装に用いたハードウェアコンポーネントを表 4 に、ソフトウェアコンポーネントを表 5 に示す。

表 4 ハードウェアコンポーネント

Table 4 Hardware Components

メモリ	CPU	GPU	コア数
64 GB	i-Core i7 8700K	NVIDIA GTX1080Ti	3584

表 5 ソフトウェアコンポーネント

Table 5 Software Components

コンポーネント	コンポーネント名	バージョン
OS	Ubuntu	18.04
実装言語	Python	3.6.4
深層学習フレームワーク	Chainer	4.2.0
評価結果の可視化	ChainerUI	0.2.0
データ加工ツール	pandas	0.20.3
可視化フレームワーク	Dash	0.30.0
グラフデータベース	Neo4j	3.4.7

6.2 プロトタイプのアーキテクチャ

提案方法を評価するために実装したプロトタイプのアーキテクチャを図 7 に示す。提案プロセスの [d] フィーチャ設計プロセス, トレーニングデータとテストデータの生成, [e] 学習モデルの生成, [f] 学習モデルの評価, [g] 学習モデルグラフの生成, [f] 学習モデルグラフの分析における処理を実装し、自動化した。

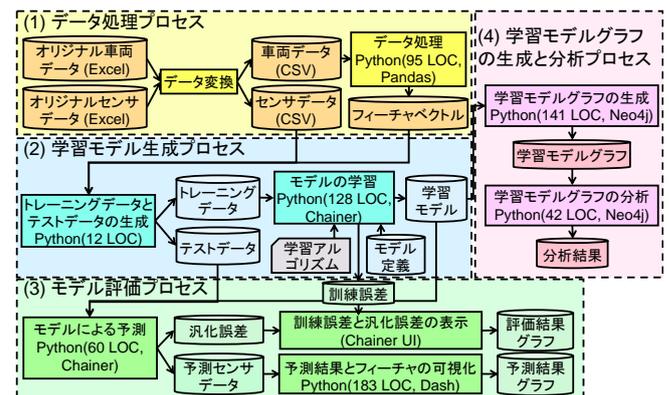


図 7 プロトタイプのアーキテクチャ

Figure 7 Prototype Architecture

(1) データ処理プロセス

取得したデータを機械学習モデルの入力(フィーチャ)として変換する処理を Python と Pandas を用いて実装し自動化した。

(2) 学習モデル生成プロセス

トレーニングデータとテストデータの生成, モデルの学習処理を Python と Chainer[16]を用いて実装し自動化した。

(3) モデル評価プロセス

予測結果の可視化を Python と Dash を用いて実装し自動化した。また、訓練誤差と汎化誤差による評価結果の可視化を ChainerUI[16]を用いて行った。

(4) 学習モデルグラフの生成と分析プロセス

学習モデルから学習モデルグラフを生成する処理と生成したグラフを分析する処理を実装した。実装言語は Python, グラフデータベースは Neo4j を用いた。

6.3 プロトタイプの実行

(1) データ処理プロセス

後述する適用では、Excel で提供されたオリジナルデータから変換したcsv形式のデータを用いてフィーチャベクトルを生成する。

(2) センサ学習モデル生成プロセス

フィーチャベクトルとラベルとなるセンサデータからトレーニングデータとテストデータを生成する。トレーニングデータ、学習アルゴリズム、モデル定義からモデルの学習をし、学習モデルグラフ生成のための学習モデルデータと訓練誤差データを生成する。

(3) モデル評価プロセス

学習モデルとテストデータを用いてモデルによる予測を行い、汎化誤差データと予測センサデータを生成する。訓練誤差と汎化誤差の学習過程における推移を Chainer UI によって可視化する。ラベルデータと予測センサデータと学習に用いるフィーチャの可視化を可視化フレームワーク Dash によって行った。

(4) 学習モデルグラフの生成と分析プロセス

学習モデルデータをグラフデータベース Neo4j に挿入し学習モデルグラフを生成する。生成した学習モデルグラフから Neo4j のクエリ言語である Cypher を用いて分析に必要なデータを抽出し分析する。

7. 自動車超音波センサデータへの適用と評価

自動車に搭載されている超音波センサデータの特性のモデル化に提案方法を適用した。超音波センサは 1 台の自動車に数個から 10 数個搭載され、近距離での衝突防止や自動駐車支援などの機能に利用され、その適用は拡大している。

7.1 適用の目的

車載超音波センサ(以下、センサと略記)の特性分析は、従来、様々な条件で繰り返し測定を行い、また、シミュレーションなどを援用しているため多くの工数と期間を要していた。さらに、センサの特性を分析するためには、車種による搭載位置、路面状況などの環境条件と認識対象物の特性といった物理空間の多様な変動要因を分析する必要があり、センサの反射波電圧を予測することは困難である。回帰分析などでは認識困難である非線形性が強く、かつ広がり大きい車載超音波センサデータに対してプロトタイプを適用し、提案方法の有効性と妥当性を評価する。

7.2 適用対象

14 車種に搭載されているセンサの路面反射波のデータに対して提案方法を適用した。センシングパターンの直接波とは、超音波の発信と受信を同一のセンサが行う場合であり、間接波とは別のセンサが行う場合である。センサがドライであるとはセンサに水滴が付着していない状態であり、ウェットは水滴が付着していることを示す。後述するように、直接波 11 車種、間接波 6 車種のそれぞれ 4 パターンの合計 68 個のデータセットに適用した。

7.3 問題設定

センサが利用されている自動車の環境条件データ(搭載位置、湿度、センシングパターンなど)からセンサが受信する超音波の反射波電圧を予測するモデルを生成する問題として問題設定をした。図 8 に本稿における問題設定を示す。適用例における学習モデルをセンサ学習モデルと呼ぶ。

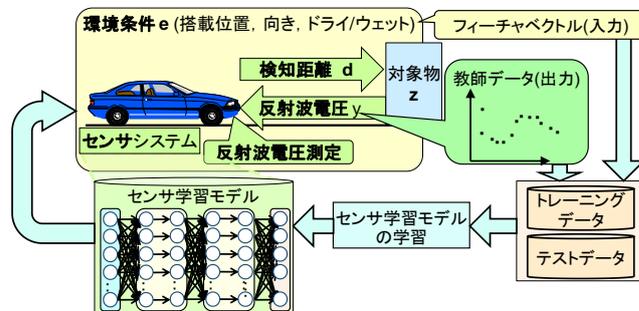


図 8 問題設定

Figure 8 Problem Context

7.4 仮説設定

センサ搭載位置データからフィーチャとして抽出すべきデータの仮説を設定する。直接波の場合は 6 個のフィーチャ、間接波の場合は発信と受信 2 つのセンサの相互関係があるため 12 個のフィーチャを抽出する仮説を設定した。

7.5 フィーチャ設計

7.5.1 探索的データ分析

入力となるセンサ搭載位置からのフィーチャと出力となる路面反射波電圧のデータを可視化した。可視化したデータは提供元との約束により公開できないので省略する。可視化した結果、直接波か間接波、センサがドライかウェットによって反射波電圧の分布が異なることを発見した。したがって、学習モデルの生成は直接波(ドライ)、直接波(ウェット)、間接波(ドライ)、間接波(ウェット)の 4 パターンで生成することが適切であると判断した。直接波とは、超音波の発信と受信を同一のセンサが行う場合であり、間接波とは別のセンサが行う場合である。センサの状態がドライであるとはセンサに水滴が付着しておらず乾燥した状態のことを示し、ウェットであるとはセンサに水滴が付着した状態を示す。

7.5.2 データクレンジング

データの特徴が表れている反射波電圧の最大値付近を抽出するために最大値付近以外をクレンジングした。センサ学習モデルが出力するデータ数を表 6 に示す。

表 6 センサ学習モデルの出力データ数

Figure 6 # of Outputs of Sensor Learning Model

	反射時間 [ms]		データ数
	min	max	
クレンジング前(全データ)	0.15	37.95	253
クレンジング後(直接波)	7.35	10.35	20
クレンジング後(間接波)	7.35	14.85	50

7.5.3 フィーチャ選択

センサの搭載位置データから直接波と間接波で反射波電圧に関係すると推定されるフィーチャを抽出し、モデルの入力とするフィーチャベクトルとして再構成した。作成したフィーチャベクトルの属性を表7に示す。コーナ(cor.)とセンタ(cen.)はセンサの位置を示している。

表7 フィーチャベクトルの属性
 Table 7 Properties of Feature Vector

直接波	間接波
高さ位置, 横位置, 天井までの高さ, 上下角, 水平角	高さ位置(cor.), 横位置(cor.), 天井までの高さ(cor.), 上下角(cor.), 水平角(cor.), 高さ位置(cen.), 横位置(cen.), 天井までの高さ(cen.), 上下角(cen.), 水平角(cen.)

7.6 センサ学習モデルの生成

学習モデルの設定を表8に示す。epochとは全てのデータを何回取り込み学習するか学習数である。本稿では、全ての学習において200 epochで収束することを確認した。バッチサイズは、1回のパラメータ修正のために取り込むデータ数を示している。最適化アルゴリズムには MomentumSGD (Stochastic Gradient Descent)を採用した。MomentumSGDとは確率的勾配降下法に慣性項を追加したことによって、より早く学習が収束することが期待できる最適化アルゴリズムである。入力層から隠れ層への活性化関数には一般に用いられているシグモイド関数を用いた。しかし、出力層にはシグモイド関数の値域を超える値を出力する必要があるためランプ関数を用いた。学習率(Learning Rate)とは、勾配法において1回の重みの更新でどれだけモデル中の重みを変化させるかを決定するハイパーパラメータである。一般的に学習の進行に応じて減少させることが望ましい。本稿では学習率の初期値を0.01とし、epochが半分を超えた時に初期値の0.1倍である0.001と更新するようにした。

表8 モデル設定

Table 8 Model Configuration

epoch	200
バッチサイズ	4
最適化アルゴリズム	MomentumSGD
活性化関数	シグモイド関数, ランプ関数
学習率	0.01, 0.001(epoch>150)
入力層のノード数	50
中間層のノード数	50
出力層のノード数	30

7.7 フィーチャの影響分析

直接波と間接波の場合の4つの主要なフィーチャの重みの平均変化率を、それぞれ、図9と図10に示す。重みの平均変化率はフィーチャの影響の大きさを表すことから、何れの場合も上下角が最大の影響力を示すフィーチャとして特定できた。直接波(5-10 epoch)と間接波(25-30 epoch)でピークを示すepoch数に差がある。これは直接波(5個)よりも間接波(10個)の入力

フィーチャ数が多く、学習の進行が遅かったことに起因すると推定される。

センサの上下角がその反射波電圧の特性に大きく影響することは想定できていたが、データから上下角がフィーチャとして影響が大きいことを明らかにしたことは本研究が初めてである。これは、提案した機械学習によるフィーチャの影響分析が有用であることを示唆している。

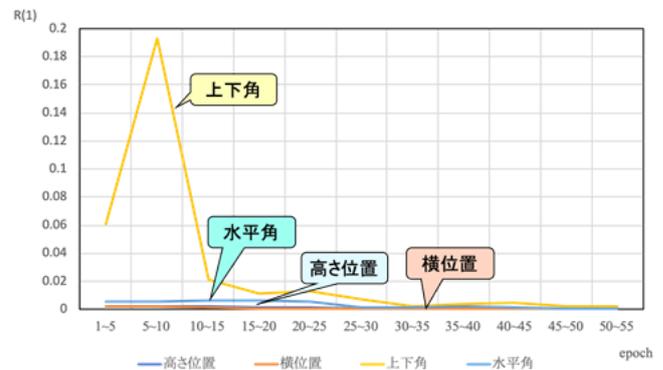


図9 重みの平均変化率(直接波)
 Figure 9 Mean Rate of Change of Weights (Direct Wave)

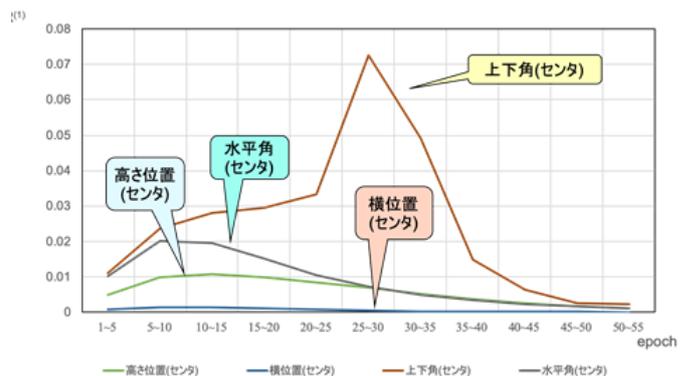


図10 重みの平均変化率(間接波)
 Figure 10 Mean Rate of Change of Weights (Indirect Wave)

7.8 認識精度の評価方法

評価関数として式(4)の平均二乗誤差を採用し訓練誤差と汎化誤差を評価した。 n は入力データ数、 f_i は*i*番目の入力 x_i に対するモデルの出力、 y_i は*i*番目の入力 x_i に対する教師データの反射波電圧を表す。

$$MSE = \frac{1}{n} \sum_{i=1}^n (f_i - y_i)^2 \quad (4)$$

7.9 認識精度の評価

センサ学習モデルの認識パターンを表9に示す。間接波の場合、センサの搭載位置にコーナのデータが含まれていないものがあるので直接波よりも認識パターン数が5つ少なくなっている。評価方法にしたがいセンサ学習モデルの認識精度を評価した結果を図11と図12に示す。4パターンの訓練誤差と汎化誤差ともに150 epoch付近で収束しており、一定の精度でモデル化可能であることを確認した。

表 9 認識パターン
 Table 9 Patterns of Recognition

センシングパターン	搭載車種	パターン数
直接波	A, D, E, F, G, I, J, K, N, O, P	11
間接波	A, B, F, H, M, N	6

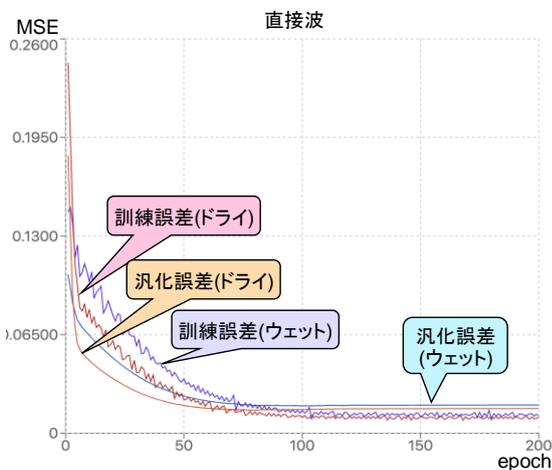


図 11 訓練誤差と汎化誤差(直接波)

Figure 11 Training Err. and Generalization Err. (Direct Wave)

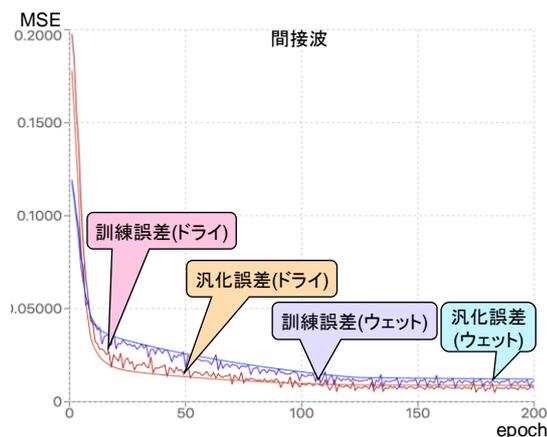


図 12 訓練誤差と汎化誤差(間接波)

Figure 12 Training Err. and Generalization Err. (Indirect Wave)

表 10 クレンジング前後の訓練誤差, 汎化誤差(直接波)
 Table 10 Training Err. and Generalization Err. of before and after Cleansing (Direct Wave)

パターン 1: 直接波の誤差		ドライ	ウェット
クレンジング前 (全データ)	訓練誤差	0.0028	0.0049
	汎化誤差	0.0034	0.0321
クレンジング後	訓練誤差	0.0015	0.0046
	汎化誤差	0.0027	0.0461

表 11 クレンジング前後の訓練誤差, 汎化誤差(間接波)
 Table 11 Training Err. and Generalization Err. of before and after Cleansing (Indirect Wave)

パターン 1: 間接波の誤差		ドライ	ウェット
クレンジング前 (全データ)	訓練誤差	0.0018	0.0020
	汎化誤差	0.0030	0.0024
クレンジング後	訓練誤差	0.0010	0.0015
	汎化誤差	0.0023	0.0024

8. 考察

8.1 学習モデルグラフによる学習のモデル化

本稿で提案した学習グラフモデルはグラフモデルの中でも表現能力の高いプロパティグラフを用いて定義することによって、分析に必要なデータを捨象することなくニューラルネットワークの構造と学習過程を表現できる。クエリによりサブグラフを取り出し可視化することで設計した学習モデルを分析することが可能になり、学習モデル設計の効率化が期待できる。本稿で提案した学習モデルグラフと学習のモデル化は著者の知る限り機械学習モデル化の新たな方法として貢献するものである。

8.2 学習モデルグラフの分析と反復学習プロセス

学習モデルグラフ上で特定した学習に影響を与えるフィーチャに基づき、フィーチャ選択の検証が可能になる。従来の試行錯誤を伴う発見的なアプローチにより多大な時間を要しているフィーチャ設計を効率化可能になる。

さらに、フィーチャ分析に基づく反復学習プロセスにより学習を一定の水準で制御可能になると期待できる。これによって、フィーチャ設計とそれを用いた反復プロセスによる機械学習モデル生成がソフトウェア工学と同様構造化された工学的方法となることが期待できる。

8.3 実データへの適用による有効性と妥当性評価

(1) 影響力が強いフィーチャの特定

車載超音波センサの直接波と間接波という性質が異なる反射波電圧データに対して、学習に対して影響力が強い支配的なフィーチャを特定可能であることを確認した。

(2) 学習モデルの認識精度

特定したフィーチャから一定の精度で学習モデルを生成可能であることを確認した。フィーチャの影響力を分析することによって、特定したフィーチャを中心にモデル化することが可能である。

8.4 先行研究との比較

先行研究[7, 8]ではモデルの出力結果となる認識精度からフィーチャの影響度を分析している。しかし、入力フィーチャによっては認識精度に現れないフィーチャの影響がある。本稿では学習モデルグラフ上でのグラフの局所的な構造分析によって影響力が強いフィーチャを特定した。しかし、特定した影響力が高いフィーチャが認識精度向上に役立っているかどうかの判定には認識精度の分析と合わせてさらに詳細な分析が必要になる。また、PCA (Principal Component Analysis)[18]や Autoencoder [18]などによって次元削減を行い、フィーチャを得る方法もあるが、どのフィーチャが重要なのか説明する必要がある場合には用いることができない。

9. 今後の課題

今後の課題は以下の3点である。

(1) 認識精度との比較分析の検討

影響力が強いフィーチャとして特定したフィーチャが学習に対して貢献するのか阻害するのか認識精度からの分析を検討

する。

(2) プロパティグラフの特性を活かした分析方法の検討

本稿で活用していないデータを用いた新たな視点からの分析を検討する。

(3) 他データ, 他の学習モデルへの適用

テーブルデータ以外のデータと再帰的な構造や畳み込み構造などのより複雑な構造の学習モデルへの適用を検討する。

10. まとめ

本稿では学習モデル構造を学習モデルグラフとしてモデル化しフィーチャの学習への影響力を分析することによる学習モデルグラフ上での仮説検証に基づく反復プロセスによる機械学習モデル生成方法を提案した。学習に影響を与えるフィーチャの特定とそれを用いた反復プロセスによって、試行錯誤を伴う発見的なアプローチによる学習モデル開発における多大なコストの低減と学習プロセスの制御が可能になる。提案方法は機械学習システムの開発プロセスがソフトウェア開発プロセスと同様な工学的プロセスとなる一アプローチとして期待できる。

謝辞

本研究を遂行するにあたり、センサデータをご提供頂いた株式会社デンソーの関係各位に感謝する。また、本研究は JSPS 科研費 JP18K11251 の助成を受けたものです。

参考文献

- [1] S. Banerjee, et al., Automation of Feature Engineering for IoT Analytics, Proc. of AIO-TAS'17, ACM, Vol. 15, Mar. 2018, pp. 24-30.
- [2] C. Borgelt, et al., Graphical Models: Representations for Learning, Reasoning and Data Mining, 2nd ed., Wiley, 2009.
- [3] G. Dong, et al., Feature Engineering for Machine Learning and Data Analytics, CRC Press, 2018.
- [4] I. Goodfellow, et al., Deep Learning, MIT Press, 2016.
- [5] Google Inc., TensorFlow, <https://www.tensorflow.org/>.
- [6] S. Jouili, et al., An Empirical Comparison of Graph Databases, Proc of SocialCom'13, IEEE, Sep. 2013, pp. 708-715.
- [7] U. Khurana, et al., Feature Engineering for Predictive Modeling Using Reinforcement Learning, Proc. of AAAI-18, Feb. 2018, pp. 3407-3414, <https://aaai.org/Library/AAAI/aaai18contents.php>.
- [8] P. W. Koh and P. Liang, Understanding Black-box Predictions via Influence Functions, Proc. of ICML '17, Vol. 70, ACM, Aug. 2017, pp. 1885-1894.
- [9] J. Li, et al., Feature Selection: A Data Perspective, ACM Computing Survey, Vol. 50, No. 6, Dec. 2017, 45 pages.
- [10] 元田 浩, 鷲尾 隆, 機械学習とデータマイニング, 人工知能学会誌, Vol. 12, No. 4, Jul. 1997, pp. 505-512.
- [11] Neo Technology, neo4j, 2016, <http://neo4j.com/>.
- [12] I. Robinson, et al., Graph Databases, 2nd ed., O'Reilly, 2015.
- [13] M. A. Rodriguez, et al., Constructions from Dots and Lines, Bulletin of the American Society for Information Science and Technology, Vol. 36, No. 6, Aug./Sep. 2010, pp. 35-41.
- [14] S. Ozdemir, et al., Feature Engineering Made Easy, Packt, 2018.
- [15] A. Petermann, et al., Graph Mining for Complex Data Analytics, Proc of ICDMW, IEEE, Dec. 2016, pp. 1316-1319.
- [16] Preferred Networks, Chainer, <https://chainer.org/>.
- [17] 白崎 悠太, 他, セマンティックグラフモデルを用いたステークホルダ分析方法の提案と評価, SES 2017 論文集, 情報処理学会, Aug.-Sep. 2017, pp. 98-105.
- [18] S. Suthaharan, Machine Learning Models and Algorithms for Big Data Classification, Springer, 2016.
- [19] The Linux Foundation, JanusGraph, 2017, <http://janusgraph.org/>.