日本ソフトウェア科学会第 36 回大会 (2019 年度) 講演論文集

ファズ・データセットを用いたメタモルフィック・テスティング ～ 機械学習ソフトウェアの検査 ～

中島 震

機械学習ソフトウェアの品質は，学習時に用いる訓練データセットに大きな影響を受ける．ソフトウェア・テスティングからみると，訓練データセットは学習ソフトウェアへのテスト入力であり，適切なデータセットを作成する方法が重要な役割を果たす．本稿では，学習プログラムを対象としたメタモルフィック・テスティング手法を基本とし，例外系テスティングで用いるファズ・データセットの系統的な生成方法を提案する．

Choice of training dataset has much impact on machine learning software quality. As such a dataset can be, from a viewpoint of software testing perspectives, regarded as a test input to a machine learning program, generating appropriate dataset plays a key role in the testing activities. This paper proposes a new metamorphic testing method with semi-randomly generated *fuzz* datasets. Generating such datasets is dependent on distortions in trained learning models.

1 はじめに

機械学習ソフトウェア技術の進展 [8] [9] と共に，極めて高い信頼性（Extreme Reliability）が必要な応用サービスへの適用が期待されるようになってきた．一方で，道路上の対象認識を行う機械学習ソフトウェアの不具合が，自動運転車による人身事故を引き起こした可能性が指摘された [29]．機械学習ソフトウェア品質評価方法の確立が急務となっている [16]．

機械学習ソフトウェアには，その品質を考える上で，従来のソフトウェアと異なる難しさがある [14]．第 1 に，品質を議論する対象が，訓練データセットを入力し訓練済み学習モデルを求める学習プログラムと，得られた訓練済み学習モデルが機能振舞いを規定する予測プログラムの 2 つに分かれている．第 2 に，計算結果が正しいか否かを判断する基準が明らかでなく，オラクル問題 [3] への対応を考える必要がある．特に，学習プログラムは，訓練結果の正解値を予め知ることが困難であり，テスト不可能プログラム [26] に分類される．

オラクル問題への解決アプローチとして，メタモルフィック・テスティング（Metamorohic Testing, MT）[5] [6] が有力な方法である．実際，深層ニューラル・ネットワーク（DNN）などの機械学習ソフトウェアの品質検査に適用する研究が進み，学習プログラム [12] [15] [27] ならびに予測プログラム [23] [28] [29] の検査事例が報告されている．検査効率の良いテスト入力の自動生成が技術的な関心の中心である．

本稿では，ニューラル・ネットワークの学習プログラムを対象として，想定外の状況下での検査での利用を考えたファズ・データセットの自動生成と，このファズ・データセットを用いるメタモルフィック・テスティングの方法を報告する．データセット多様性 [13] を示すテスト入力の具体的な方法であって，訓練済み学習モデルの歪み [17] に着目するものである．

2 機械学習ソフトウェアの特徴

ニューラル・ネットワークの教師あり分類学習を具体例として機械学習ソフトウェアの特徴を整理する．

Metamorphic Testing with Fuzz Dataset.

Shin Nakajima, 国立情報学研究所, National Institute of Informatics.

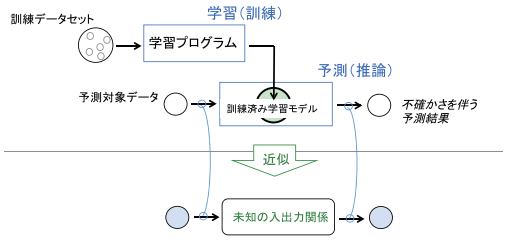


図 1　Algorithmic Modeling

2.1　入出力マッピング

図 1 に示すように，DNN は未知の入出力関係を関数近似で具体的に表現したネットワーク計算モデル [9] である．入力の対象データから予測結果の出力値を計算する手順を表すことから，アルゴリズミック・モデリング [4] と云われる．

入出力関係が複雑なことから，すべての場合を事前に分析し，プログラム化することが難しい．そこで，膨大な規模の訓練データセットから，DNN モデルを訓練学習する方法を用いる．以下，教師あり分類学習タスクを想定して，訓練学習の基本的な方法を説明する．多次元ベクタとして表されたデータを C 個のカテゴリ（クラス）に分類する学習タスクである．

多次元データ・ベクタ $\vec{x}^n$ と正解タグ t^n の組の集まりをデータセット DS とし，$DS = \{\langle \vec{x}^n, t^n \rangle\}$ と表す．データ $\vec{x}$ を入力としパラメータ W を持つ非線形関数を $y(W;\vec{x})$ とする．W を学習パラメータと呼び，その値が確定して W^c とする時，$y(W^c;\vec{x})$ は入力 $\vec{x}$ に対するタグ値の予測結果を返す．$y(W^c;\vec{x})$ は求めた未知の入出力関係（図 1）の良い近似となる．

なお，パラメータ W の値が未定の時，$y(W;\_)$ の全体は，W を指標とする関数の集まりを表し，学習モデルと呼ばれる．

2.2　最適化問題

学習は，学習モデルを決めた後，与えられた訓練データセット $LS = \{\langle \vec{x}^n, t^n \rangle\}$ に対して，誤差関数 $\mathcal{E}$ を最小化するパラメータ値 W^* を求める数値最適化である．

$$W^* \;=\; \underset{W}{argmin}\; \mathcal{E}(W;\{\langle \vec{x}^n, t^n \rangle\})$$

LS を入力値として，この最適化問題を解く学習プログラムを $\mathcal{L}_f(LS)$ と表記する．今，$\ell(y(W;\vec{x}^n),\ t^n)$ を，$\vec{x}^n$ に対する予測計算値 $y(W;\vec{x}^n)$ と正解タグ t^n の距離とする時，LS の大きさを N として，誤差関数 $\mathcal{E}$ を次のように定義する．

$$\mathcal{E}(W;\{\langle \vec{x}^n, t^n \rangle\}) \;=\; \frac{1}{N}\sum_{n=1}^{N} \ell(y(W;\vec{x}^n), t^n)$$

この誤差関数 $\mathcal{E}$ の最小化は，LS の経験分布 ρ_{em} に対する ℓ の期待値を最小化する W^* を求める最適化問題

$$W^* \;=\; \underset{W}{argmin}\; E_{\langle \vec{x},t \rangle \sim \rho_{em}} [\![\ell(y(W;\vec{x}), t)]\!]$$

である．

上記で求めた学習パラメータの最適解 W^* を用いて，運用時の入力データ $\vec{x}$ に対して予測結果を返す推論プログラム $\mathcal{I}_f$ を，W^* あるいは $y(W^*;\vec{x})$ を用いて定義する．分類学習タスクでは，データ $\vec{x}^m$ に対する $\mathcal{I}_f$ の計算結果を，C 個のカテゴリに属する確からしさを表す確率分布 $Prob(\vec{x}^m, c)$ とする．

機械学習ソフトウェアの品質を論じる対象は，$\mathcal{L}_f$ および $\mathcal{I}_f$ の 2 つがある．$\mathcal{L}_f$ は，用いた学習方式を実現した数値計算プログラムであって，上位仕様としての学習方式を忠実にプログラム化しているか否かが品質に関わる．$\mathcal{I}_f$ は予測性能が品質を論じる観点となる．また，予測性能は LS が含まない入力データに対する予測結果を基本とする．この予測結果は $\mathcal{L}_f$ の計算結果である訓練パラメータ W^* によって決まる．したがって，$\mathcal{L}_f$ および $\mathcal{I}_f$ 共に，その品質に関わる議論は訓練データセット LS に依存する．

2.3　収束性と正解率

ニューラル・ネットワークの最適解探索は非凸問題であり，解への収束が保証されていない．また，訓練データセット LS に過適合，つまり，過学習の状況は，期待する W^* として好ましくない．収束性の向上と過学習の軽減が学習アルゴリズムの鍵となる [9]．

通常，繰り返し探索過程（エポック進行過程）で，収束性と正解率の推移を監視することで，学習過程を観測する．エポック e 時点でのパラメータ値を W^e とする時，収束性は誤差関数の値 $\mathcal{E}(W^e;LS)$ である．また，データセット $DS = \{\langle \vec{x}^m, t^m \rangle\}$ の W^e に対す

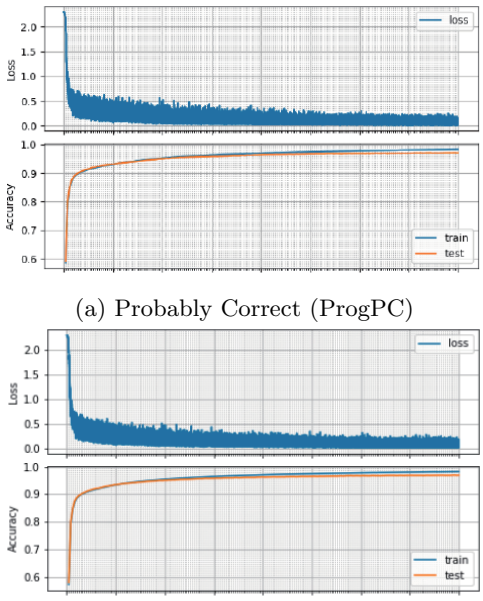
る正解率を

$$acc(W^e; DS) = \sharp(arg\max_{c \in C} Prob(\vec{x}^m, c) = t^m)$$

とする．ただし，$\sharp P(x)$ は，有限集合 X の要素 x について述語 $P(x)$ が論理真となる頻度を表す．LS と異なるデータセット TS を試験データセットとして，エポック進行過程で $acc(W^e; LS)$ ならびに $acc(W^e; TS)$ を監視する．2 つが同等であれば，過学習が軽減されている．

図 2 は収束性と LS および TS に対する正解率をエポック進行に沿ってグラフ化した．ただし，LS と TS は同じ経験分布を持つ異なる標本である．後の議論と関係することから，図 2(a) と (b) に 2 つのプログラムに対するグラフを示す．図 2(a) は標準的な学習方法を恐らく正しく実現したプログラム（ProgPC[†1]）であり，図 2(b) は前記に欠陥を意図的に混入したプログラム（ProgBI[†2]）である．

図 2 から，エポック進行と共に，収束傾向が見られること，ならびに，LS と TS の正解率が同等であり共に良い値（95% 以上）になっていること，がわかる．また，図 2(a) と (b) のグラフを区別することは難しく，プログラム欠陥の有無が判断できないことがわかる [†3]．



(a) Probably Correct (ProgPC)

(b) Bug-Injected (ProgBI)

図 2 Loss and Accuracy: MNIST dataset

3 ソフトウェア・テスティング

ソフトウェア・テスティングでは，何をテスト入力とするか，どのようにして実行結果が正しいかを判定するか，という 2 つの観点が重要になる [2]．前者をテスト入力生成問題，後者をオラクル問題と呼ぶ．

3.1 ファズ・テスティング

最初に，テスト入力生成の問題に関連する事項を説明する．プログラムの品質を確認する際，正常系テスティング（Positive Testing）と例外系テスティング（Negative Testing）を適宜行う．

正常系テスティングは，検査対象が想定した機能振舞いを示すことの確認である．説明の都合上，プログラムが，事前・事後条件を伴うとしよう．プログラム本体 $Body$ と事前条件 Pre，事後条件 $Post$ の間には，「事前条件を満たす時，プログラム本体実行後の状態で事後条件を満たす」という関係が成り立つ．事前条件 Pre を満たすテスト入力データに対して，上記の関係が成り立つ時，プログラム $Body$ はテストに合格したとする．

例外系テスティングは，想定外の状態でプログラムが破滅的な不具合を生じないことを確認する．一般に，プログラムは Pre を満たさない入力に対しても，それなりの振舞いを示すことが期待される．$\neg Pre$ を満たすデータが入力された場合に暴走して異常終了するようなことがあってはならない．従来から，システム・ソフトウェアやセキュリティ・システムなどのオープンなソフトウェアの例外系テスティングでは，ランダムに生成したデータをテスト入力とするファズ・テスティング（Fuzz Testing）の方法 [11] が知られている．

†1 Probably Correct.
†2 Bug-Injected.
†3 有無がわからないように欠陥を混入することが可能である．

3.2 メタモルフィック・テスティング

メタモルフィック・テスティング（MT） [5] は，数値計算あるいはコンパイラを含むトランスレータなど，入力に対する計算結果の正解値を予め知ることが難しいテスト不可能プログラム [26] の検査方法として考案され，その後，暗黙のオラクルを用いるシステム・ソフトウェアやセキュリティ・システムの検査に適用された．機械学習を含む幅広いソフトウェアのテスティング方法論として使われるようになってきた．ここでは，文献 [6] をベースに MT の概要を紹介する．

MT は検査対象に適切したメタモルフィック関係（Metamorphic Relations, MR）を用いるソフトウェア・テスティングの方法論である．今，検査対象を関数 $f : D_1 \to D_2$ で表す．N 個（$N \geq 2$）の入力 $x^{(n)}$ ($1 \leq n \leq N$) と N 回のテスト実行結果 $f(x^{(n)})$ に対する $2N$ 項関係 $\mathcal{R}(x^{(1)}, \ldots, x^{(n)}, f(x^{(1)}), \ldots, f(x^{(n)}))$ ($\mathcal{R} : D_1^N \times D_2^N$) を MR と呼ぶ．

多くの MT 適用事例と同様に本稿の方法は，上記の一般的な MR を制限した形で用いる．MR をプログラムへの入力データの関係 T と出力に関わるメタモルフィック関係 Rel_T に分解する．以下，$N = 2$ の場合について説明する．

検査対象 f の 2 つの異なる実行結果の間で満たすべき関係を $Rel_T : D_2 \times D_2$ とする．Rel_T は f の機能振舞いに関わる上位仕様から導出されると考える．この時，次のような関数 $T : D_1 \to D_1$ があるとする．

$$\exists T \in D_1 \to D_1 \,.\, \forall x \in D_1 \,.\, Rel_T(f(x),\ f(T(x)))$$

変換関数 T によって生成されたデータをフォローアップ・テスト入力と呼ぶ．MT は，このような変換関数 T とメタモルフィック関係 Rel_T を用いて，

$$\exists x' \in D_1 \,.\, \neg Rel_T(\ f(x'),\ f(T(x'))\ )$$

となる x' を見つけるテスティング法である．見つかった場合，f に欠陥があると推定する．見つからなければ，テスト範囲で欠陥の有無は不明である．

MT の方法では，$f(x)$ と $f(T(x))$ の 2 つのプログラム実行結果が関わる．いずれも，設計仕様や理論的な考察から得られる正解値ではないことから，通常のオラクルとはならない．互いに相対的な正解の役割を果たす黄金出力（Golden Outputs）と呼ぶプログラム実行結果であり，部分オラクル（Partial Oracles）と云われる．

3.3 データセット多様性

部分オラクルの役割を果たす黄金出力を作り出す方法として，従来は，デザイン多様性（Design Diversity）とデータ多様性（Data Diversity）の 2 つが知られていた [1]．デザイン多様性は，N-バージョン・プログラミングで実現されることが多く，同じ機能仕様を満たすプログラムを複数（N 個）開発し，同じテスト入力に対して同じ出力を得るかを確認する．データ多様性は，検査対象が満たす性質から，テスト入力に用いる多様なデータを系統的に求める方法論である．

機械学習ソフトウェアは，膨大なデータの集まり（データセット）から訓練済み学習モデルを導出し（$\mathcal{L}_f$），そのモデルを用いて運用実行時に予測推論結果を返す（$\mathcal{I}_f$）．いずれにしても，機械学習ソフトウェアの品質はデータセットに依存する．そこで，テスト入力としてのデータセットに着目した検査法を考える必要がある．

データセット多様性（Dataset Diversity）[13] は，学習プログラム $\mathcal{L}_f$ のテスティングを行う際に，さまざまなデータセットを用いるという考え方である．機械学習では，与えた訓練データセット LS が，予測推論プログラム $\mathcal{I}_f$ の機能を内在するとみなしていた．LS が暗黙に想定する機械学習タスクとの関連を考えなくてはならない．

正常系テスティングは，LS と同じ経験分布を持つが LS と異なる標本（データセット）をテスト入力とする．一方，例外系テスティングであっても，従来のファズのようにランダム生成したデータセットは役立たない．このようなデータセットは LS と全く異なる学習タスクになる可能性が大きいからである．

そこで，訓練データセット LS を初期データセット $DS^{(0)}$ とし，LS と少し異なるデータセット $DS^{(K)}$ を順次生成する．$DS^{(K)}$ を求める際，$DS^{(K-1)}$ に対する学習結果の情報を用いる．たとえば，検査対象が SVM の場合は，$DS^{(K-1)}$ に対して求めた分離ハイパー面の近傍に新たなデータを追加した $DS^{(K)}$ のテスティング効果が大きかった [12]．そこで，目的に

合った変換関数 T' を定義し，

$$DS^{(K)} = T'(DS^{(K-1)}, g(DS^{(K-1)}))$$

によって，順次，異なるデータセットを生成する．先の SVM の例では，g として，訓練データセット $LS^{(K-1)}$ に対する学習結果 $\mathcal{L}_f(LS^{(K-1)})$ を選ぶ場合に相当する．この関係式を書き直して，$T(LS^{(K)})\equiv T'(LS^{(K)}, \mathcal{L}_f(LS^{(K)}))$ とすれば，$LS^{(K)} = T(LS^{(K-1)})$ であり，MT 法の定義と良く対応する．

4 ファズ・データセットによるテスティング

訓練済み学習モデルの歪み [17] に着目したメタモルフィック・テスティングの方法を説明する．

4.1 歪みデータセット

MT のフォローアップ入力生成法として，敵対データ例の生成方法 L-BFGS [22] を利用した「歪みデータ」の作成法を示す．

元データ $\vec{x}_S$ をラベル t_T に誤分類させる敵対データ例は制約条件付きの最適化問題の解 $\vec{x}^*$ として求めることができる．W^* を訓練済み学習パラメータの集まりとして，

$$\begin{aligned} & A_\lambda(W^*; \vec{x}_S, t_T, \vec{x}) \\ =\ & \ell(\vec{y}(W^*;\ \vec{x}),\ t_T) + \lambda\ell(\vec{x},\ \vec{x}_S) \end{aligned}$$

に対して，

$$\vec{x}^* = \underset{\vec{x}}{argmin}\ A_\lambda(W^*; \vec{x}_S, t_T, \vec{x})$$

$$\text{s.t. } |x_j| \leq 1$$

によって求める．多次元データ $\vec{x}$ を $\langle \ldots, x_j, \ldots \rangle$ とした．関数 A_λ の定義式中，第 1 項は $\vec{x}$ に対する予測結果が与えられた教師タグ t_T を，第 2 項は $\vec{x}$ が指定のデータ $\vec{x}_S$ を再現することを表す．

上記の方法を利用して次のような変換関数を考える．入力データセット LS の要素である元データ $\vec{x}^n$ 全てにノイズを挿入したデータセットを求める．

$$\vec{x}^{(n)*} = arg\ \underset{\vec{x}}{min}\ A_\lambda(W^*; \vec{x}^n, t^n, \vec{x})$$

として，

$$T_A(LS, W^*) = \{\ \langle \vec{x}^{(n)*},\ t^n \rangle\ \}$$

は，歪んだ入力ベクトル $\vec{x}^{(n)*}$ からなるデータセットとなる．λ が微小値の場合，歪みの度合いが大きい．λ が適切な大きさの時，$\vec{x}^{(n)*}$ はノイズの影響が小さく，$\vec{x}^{(n)}$ とほぼ一致する．次に上記の T_A を用いて，

$$T_\lambda(LS^{(K)}) = T_A(LS^{(K)}, \mathcal{F}_L(LS^{(K)}))$$

と定義すると，$LS^{(K)} = T_\lambda(LS^{(K-1)})$ と表せる．ただし，$LS^{(0)}$ は与えられた訓練データセットとする．さまざまな λ 値を与えたり，あるいは，ある特定の λ 値に対する T_λ を繰り返して適用したりすることで得られるデータセットを LS_λ と総称する．LS_λ は，LS と同じ学習タスクを表す一方で，異なる経験分布を持つ．つまり，データセット多様性を示す多数のデータセットを系統的に生成することができる．

4.2 検査の指標

前節の T_λ をフォローアップ入力生成関数とする場合について，適切なメタモルフィック関係を求める．正解率関数 acc を用いる方法を考える．以下，$W^{(K)*} = \mathcal{L}_f(LS^{(K)})$ と略記する．

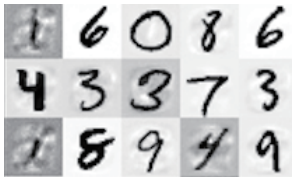
先の最適化問題の形から，特別な場合として，$\vec{x}^n$ は $\vec{x}^{(n)*}$ になり得る．つまり，$LS^{(1)}$ は $LS^{(0)}$ と「同等な」分類学習タスクを定義すると考えて良い．一方，λ 値の選び方によって，$W^{(0)*}$ に適合する範囲内で，$\vec{x}^{(n)*}$ の $\vec{x}^n$ からの「ズレ」が生じる．A_λ からわかるように，$W^{(0)*}$ に過適合するように $\vec{x}^{(n)*}$ を決めることができる．

歪みデータセットを繰り返し生成する場合を考えると，$W^{(K)*}$ は $LS^{(K+1)}$ への適合度が高く，仮想的な極限では $LS^{(\infty)}$ に「最適化」される．その結果，$LS^{(0)}$ と同じ経験分布の試験データセット TS に対する正解率が低下するが，K が異なっても $W^{(K)*}$ の TS に対する正解率は大きな変化がないと期待できる．

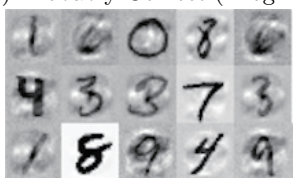
簡単のため，収束と判定されたエポックでの正解率を考え，$ACC(DS, TS) = acc(\mathcal{L}_f(DS), TS)$ とおく．$LS^{(K)}$ に対して，微小値 ϵ を選んで，

$$\begin{aligned} & Rel_T(LS^{(K)}, LS^{(0)}) \\ =\ & |ACC(LS^{(K)}, TS) - ACC(LS^{(0)}, TS)| \leq \epsilon \end{aligned}$$

を MR としよう．つまり，$LS^{(0)}$（あるいは $LS^{(K)}$）の訓練結果である $W^{(0)*}$（あるいは $W^{(K)*}$）を用いて TS に対する正解率を比較する．

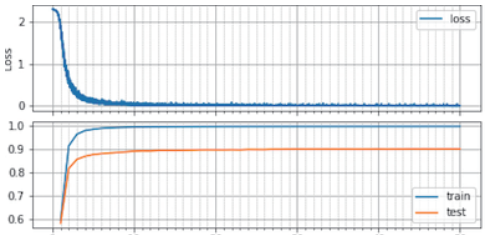


(a) Probably Correct (ProgPC)

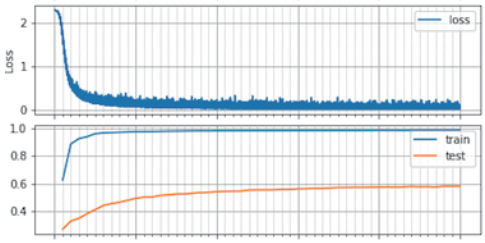


(b) Bug-Injected (ProgBI)

図 3 Distorted Data



(a) Probably Correct (ProgPC)



(b) Bug-Injected (ProgBI)

図 4 Loss and Accuracy: Follow-Up Dataset

5 実験

5.1 手書き数字分類

手書き数字分類の標準ベンチマーク問題である MNIST データセットの実験結果[18] を報告する．

実験は欠陥がないと考えられる学習プログラム（ProgPC）と欠陥を混入した学習プログラム（ProgBI）の 2 つの系列で行って結果を比較した．最初に，MNIST の訓練データセット LS と試験データセット TS を用いた収束性と正解率のグラフは図 2(a) と (b) に示す通りである．

次に，MNIST データセットの LS を $LS^{(0)}$ とし，$LS^{(K)} = T_\lambda(LS^{(K-1)})$ によって，順次，歪みデータセットを求める．図 3 に，$LS^{(1)}$ の一部を図示した†4．もとの MNIST データに何らかの「ノイズ」が被さっている．目視する限り，あきらかに，図 3(b) は図 3(a) よりも「汚れて」いる．つまり，歪みの度合いが大きい．

図 4 は $LS^{(1)}$ を訓練データセットとした時の学習過程を示す．ここで，試験データセットは TS である．図 4(a) と (b) 共に，訓練データセットに過適合（過学習）していることが確認できる．つまり，$W^{(0)*}$ が LS (1) に過適合していることを表す．一方，TS に対する正解率を比較すると，図 4(a) が 90%，図 4(b) が 60%と大きく異なる．先の MR 定義に与える ϵ を 0.1 程度に設定すると，収束と考えられるエポックで，図 2(b) では 95%を超えていたことから，図 4(b) は MR を満たさない．

以上，MT の方法によって，ProgBI に欠陥があると推測できる．

5.2 議論

欠陥の意図的な混入実験によって，欠陥の有無が図 4 のグラフ（TS に対する正解率グラフ）に大きな影響を与えることが読み取れる．欠陥がある学習プログラムは「歪んだ」学習モデルを作り，これを用いる変換関数 T_A は「大きな」歪みのデータセットを作る．その結果，訓練データセットとして用いたファズ・データセット $LS^{(1)}$ と試験データセット TS の経験分布との違いが大きくなり TS に対する正解率が低下した．

学習プログラム $\mathcal{L}_f$ に欠陥があると、訓練済み学習モデルあるいは W^* に大きな歪みを生じる [17]．この歪みを直接観察する方法があれば，$\mathcal{L}_f$ の欠陥の有無を調べることができるだろう．残念なことに，歪みを観察する方法がないので，歪みのある W^* を用いたデータセット生成関数 T_λ によってフォローアッ

†4 直感的な見やすさから，MNIST データの白黒を反転させて表示している．

プ入力となるファズ・データセット LS' を生成した．この生成プログラムに欠陥がないと仮定すると，LS' の歪みは，前記の W^* が内在していた歪みが伝播したものである．このようなデータセット LS' の経験分布は計算できるが，経験分布の違いの具体的な影響は収束性と正解率として顕在化する．経験分布も歪みを直接観察する方法ではない．

仮に，検査対象の学習プログラム $\mathcal{L}_f$ を歪みを観察する関数と同一視してみよう．上記のメタモルフィック・テスティング過程中，訓練データセット LS から W^* を得る過程と正解率の観察を目的として LS' を学習する過程で $\mathcal{L}_f$ を 2 回実行していることがわかる．欠陥が増幅することになり，TS に対する正解率の違いとして観察できたと考えられよう．

6 関連研究

機械学習ソフトウェアへのメタモルフィック・テスティングで導入された既存の入力データセット自動生成方法（あるいはデータ自動生成方法）と比較する．

文献 [27] は，K-近傍法やサポート・ベクター・マシン（SVM）の学習プログラム $\mathcal{L}_f$ 検査への応用から，分類学習に一般的なメタモルフィック関係を整理した．文献 [12] は SVM を対象とし，宣言的な問題記述から系統的にフォローアップ入力データ生成法を導出する方法を論じた．その後，ニューラル・ネットワークの $\mathcal{L}_f$ のメタモルフィック・テスティングに対して，データセット多様性 [13] を導入する方法が提案された [15]．歪みのあるデータセットをグラフ探索アルゴリズムで生成する．生成効率が良い反面，他の学習タスクや DNN 学習モデル一般に対する適用可能性の見通しがよくない．

DNN を対象とする研究は，推論プログラム $\mathcal{I}_f$ の検査が中心であり，訓練データセットが暗に規定する想定から外れた実行時入力データに対する振舞いを調べる．一般的にはデータセット・シフト（Dataset Shift） [21] の状況での検査と云えよう．各々，データの生成方法に特徴がある．

文献 [29] は与えた条件を満たすデータをランダム生成するファジングの方法を用いた．DeepTest [23] はグラフィックス処理で元データから新たなデータを生成する．これは従来からある CNN を対象としたデータ拡張（Data Augmentation）[10] の方法に準じている．特に，試験時拡張（Test-Time Augmentation）といえる．文献 [29] ならびに [23] は試験データセットの統計的な性質を考慮していない．

DeepRoad [28] は，敵対生成ネットワーク（Generative Adversarial Networks, GAN）に基づく方法によって新たなデータを生成する．ここで，GAN [24] は，零和[†5]ゲーム [19] として定式化される．与えられたデータセットの経験分布 ρ_{em} にしたがう新しいデータを生成する [7]．元の試験データセットと同様な分布にしたがうことから，「想定範囲での網羅性向上」といえる．

文献 [25] は，モンテカルロ法の重点サンプリングの考え方で，コーナーケースとなるデータを効率よく生成し，そのデータを誤推論するか否か，つまり，予測のロバスト性を統計的に評価する．「想定範囲外を含む」統計テスティング（Statistical Testing）といえる．

本稿の方法は，歪みデータを最適化問題によって生成するもので，初期訓練データセットの経験分布から外れるようなデータセット多様性を実現する．「想定から離れたデータセット」生成といえる．そこで，ファズ・データセット（Fuzz Dataset）と命名したのだった．提案方法は $\mathcal{L}_f$ の検査を目的としていたが，個々のデータ（ファズ）を使って $\mathcal{I}_f$ の検査を行うことが可能である．

7 おわりに

ニューラル・ネットワークに基づく機械学習プログラムのメタモルフィック・テスティング法として最適化問題によるデータセット多様性を導入した．訓練済み学習モデルの歪み [17] という見方を基本とする．今後，歪み度合いの精密な定義を行う必要がある．また，再帰型ニューラル・ネットワークなどへの適用検討を進めていきたい．

謝辞 本研究の一部は JSPS 科研費 JP18H03224 の

†5 「れいわ」と読み zero-sum のことである．

助成および NEDO の委託業務（NII へは再委託）によって実施した．MT 法について議論して下さった T.Y. Chen 教授（Swinburne 工科大学）ならびに実験に協力した下さった今井克則氏（(株）グラッツ）に感謝する．

参 考 文 献

[1] Ammann, P. and Knight, J.C. : Data Diversity: An Approach to Software Fault Tolerance, *IEEE TC*, vol.37, no.4, pp.418-425, 1988.

[2] Ammann, P. and Offutt, J. : *Introduction to Software Testing*, Cambridge University Press 2008.

[3] Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., and Yoo, S.: The Oracle Problem in Software Testing: A Survey, *IEEE TSE*, 41 (5), pp.507-525, 2015.

[4] Breiman, L. : Statistical Modeling: The Two Cultures, *Statistical Science*, 16(3), pp.199-231, 2001.

[5] Chen, T.Y., Chung, S.C., and Yiu, S.M. : Metamorphic Testing - A New Approach for Generating Next Test Cases, HKUST-CS98-01, The Hong Kong University of Science and Technology, 1998.

[6] Chen, T.Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T.H., and Zhou, Z.Q. : Metamorphic Testing: A Review of Challenges and Opportunities, *ACM Computing Surveys* 51(1), Article No.4, pp. 1-27, 2018.

[7] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: Generative Adversarial Nets, In *Adv. NIPS 2014*, pp.2672-2680, 2014.

[8] Goodfellow, I., Bengio, Y., and Courville, A.: *Deep Learning*, The MIT Press 2016.

[9] Haykin, S.: *Neural Networks and Learning Machines (3ed.)*, Pearson India 2016.

[10] Krizhevsky, A., Sutskever, I., and Hinton, G.E.: Imagenet Classification with Deep Convolutional Neural Networks, In *Adv. NIPS 2012*, pp.1097-1105, 2012.

[11] Miller, B.P. , Fredricksen, L. , and So, B. : An Empirical Study of the Reliability of UNIX Utilities, *Comm. ACM*, vol.33, no.12, pp.32-44, 1990.

[12] Nakajima, S. and Bui, H.N.: Dataset Coverage for Testing Machine Learning Computer Programs, In *Proc. 23rd APSEC*, pp.297-304, 2016.

[13] 中島震：データセット多様性のソフトウェア・テスティング，コンピュータ・ソフトウェア，35(2), pp.26-32, 2018.

[14] Nakajima, S.: Quality Assurance of Machine Learning Software, In *Proc. GCCE 2018*, pp.601-604, 2018.

[15] Nakajima, S. : Dataset Diversity for Metamorphic Testing of Machine Learning Software, In *Proc. SOFL+MSVL2018* pp.21-38, 2018.

[16] 中島震，妹尾義樹，大岩寛，磯部祥尚：機械学習ソフトウェアの品質評価保証レベル，電子情報通信学会ソフトウェアサイエンス研究会，那覇，2019.

[17] 中島震: モデルの歪みと機械学習プログラムの欠陥，情報処理学会 SIGSE，小樽，2019.

[18] Nakajima, S. and Chen, T.Y.: Generating Biased Dataset for Metamorphic Testing of Machine Learning Programs, In *Proc. IFIP-ICTSS 2019*, (to appear), 2019.

[19] 岡田章 :ゲーム理論（新版），有斐閣 2011.

[20] Pei, K., Cao, Y., Yang, J., and Jana, S.: DeepXplore: Automated Whitebox Testing of Deep Learning Systems, In *Proc. 26th SOSP*, pp.1-18, 2017.

[21] Quinonero-Candela, J., Sugiyama, M. Schwaighofer, A., and Lawrence, N.D. (eds.) : *Dataset Shift in Machine Learning*, The MIT Press 2009.

[22] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R.: Intriguing properties of neural networks, In *Proc. ICLR 2014*, 2014.

[23] Tian, Y., Pei, K., Jana, S., and Ray, B.: DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars, In *Proc. 40th ICSE*, pp.303-314, 2018.

[24] Warde-Farley, D. and Goodfellow, I.: Adversarial Perturbations of Deep Neural Networks, in *Perturbation, Optimization and Statistics*, The MIT Press 2016.

[25] Webb, S., Rainforth, T., Teh, Y.W., and Kumar, M.P.: A Statistical Approach to Assessing Neural Network Robustness, In *ICLR 2019*, 2019.

[26] Weyuker, E.J.: On Testing Non-testable Programs, *Computer Journal*, 25 (4), pp.465-470, 1982.

[27] Xie, X., Ho, J.W.K., Murphy, C., Kaiser, G., Xu, B., and Chen, T.Y.: Testing and Validating Machine Learning Classifiers by Metamorphic Testing, *J. Syst. Softw.*, 84(4), pp.544-558, 2011.

[28] Zhang, M., Zhang, Y., Zhang, L., Liu, C., and Khurshid, S.: DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems, In *Proc. ASE′18*, pp.132-142, 2018.

[29] Zhou, Z.Q. and Sun, L.: Metamorphic Testing of Driverless Cars, *Comm. ACM*, vol.62, no.3, pp.61-67, 2019.