統計的な部分オラクルによるテスティング方法

中島 震

ソフトウェア技術の広がりと共に，検査条件が不確かさを伴う，正解値が既知でない，といった 2 つの問題を考慮したテスティング方法の確立が期待されている．本稿では，統計的なメタモルフィク・テスティングの考え方を整理し，統計的な手法と部分オラクルを組み合わせる一般的なテスティング・フレームワークを提案する．次いで，このようなソフトウェアの代表例である機械学習プログラムのテスティングへの適用法を示し，ニューラル・ネットワークの訓練・学習プログラム検査の実験結果を報告する．

New methods of software testing are called for to take into account two issues on the test oracle problems, uncertainties of the testing conditions and unknown correctness criteria. This paper proposes a testing framework general enough to account for the existing statistical metamorphic testing methods. The framework is amenable to adapt itself to machine learning testing, which is illustrated with an experiment on testing of neural network programs.

1 はじめに

ソフトウェア・テスティング [2] は，プログラムが期待される機能振舞いを実現しているかを検査する現実的な方法である．ソフトウェア工学の主要な分野のひとつとして，さまざまな観点から基本的な技術の研究が進められてきた [24].

ソフトウェア技術の広がりと共に，新しい特徴を持つプログラムが検査対象になっている．CPS [18] やビッグデータ・アナリティックス [25] は，現実世界の現れとしてのデータを取り扱う．このようなプログラムの検査では，多様な原因から生じた不確かさ（Uncertainty）を前提とする方法を考えなければならない [8]．深層ニューラル・ネットワーク（DNN）[10] に代表される機械学習ソフトウェアのテスティング技術 [20] [33] は同様な特徴を持つ．検査対象プログラムの正しさを確認するテスト・オラクル（オラクル）[13] を定義することが難しい．オラクル問題 [3] に新たな挑戦課題をもたらしたといえる．

本稿では，不確かさを考慮したテスティングの技術を整理し，統計的なオラクルと部分オラクルを組み合わせた一般的なテスティング・フレームワークを提案する．次いで，機械学習プログラムのテスティングへの応用方法を示し，ニューラル・ネットワークの訓練・学習プログラム検査の実験結果を報告する．

2 統計的なテスティング法

テスト・オラクルの基本を紹介し，既存の統計的なテスティング法を整理する．

2.1 テスト・オラクル

ソフトウェア・テスティングでは，検査対象プログラムの出力が正しいか否かを調べるテスト・オラクル（オラクル）が存在することを仮定する．

今，簡単化して，検査対象プログラムを関数 $f \in D \to D'$ とする．オラクルは，$f(x)$ が正しいことの検査に関わる [13]．素朴には，テスト入力値 $a \in D$ に対する出力結果 $f(a) \in D'$ が正しい値 $C_a \in D'$ に一致するかを調べる方法である．述語 $\mathcal{O}_T^{[N]} \stackrel{def}{=} (C_a = f(a))$ で表す．

入力 a に対する正解値がプログラム $f(x)$ の機能仕

Software Testing with Statistical Partial Oracles.

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

様あるいは理論値 S_a から決まっている場合，仕様に基づくオラクル（Specified Test Oracle）[3] と呼ぶ．一般に，$S_a \in D'_A$ はプログラムよりも高い抽象レベルの形式で表されていることから，事前に S_a をプログラムで扱えるデータに具体化して（Concretizing）おく．C_a から S_a への抽象化関数を $abs \in D' \to D'_A$ とすると，$S_a = abs(C_a)$ の関係が成り立つ．オラクルは abs を含み，$\mathcal{O}_T^{[S]} \stackrel{def}{=} (abs^{-1}(S_a) = f(a))$ である．

仕様に基づくオラクルが存在しないプログラムの検査では，検査対象以外のプログラムが出力した結果を利用する [7]．これを導出に基づくオラクル（Derived Test Oracle）[3] と呼ぶ．N バージョン・プログラミングあるいはデザイン多様性の方法 [1] で開発したプログラム $g(x)$ の計算結果を疑似オラクル（Pseudo Oracles）として検査する．$\mathcal{O}_T^{[D]} \stackrel{def}{=} (g(a) = f(a))$ である．$f(x)$ を検査対象とする時，疑似オラクルは，$g(x)$ の開発と実行 $g(a)$ の手順を含む．特に，$g(a)$ の計算結果をゴールデン出力（Golden Outputs）と呼ぶ．

また，メタモルフィック・テスティングの方法 [5] では，入力データの変換関数（フォローアップ・テスト入力生成関数）$T \in D \to D$ に対して，述語 $R_T(f(a), f(T(a)))$ をメタモルフィック関係と呼ぶ．メタモルフィック関係が等価関係（$\_ = \_ \in D' \times D'$）の時，$\mathcal{O}_T^{[M]} \stackrel{def}{=} (f(a) = f(T(a)))$ である．

なお，$f(a)$ と $f(T(a))$ は共に，プログラムの実行結果であり，どちらか一方が正しいということはない．2 つの実行結果の整合性を調べている．そこで，$f(a)$ と $f(T(a))$ は互いに部分オラクル（Partial Oracle）[6] の関係になる．デザイン多様性の方法による疑似オラクルにおいても，ゴールデン出力 $g(a)$ の正しさは確認されているわけではない．$f(a)$ と $g(a)$ も互いに部分オラクルといえる．

さて，ここまでは，検査対象プログラムの挙動が確定的（Deterministic），つまり，同一の入力データ a に対して，$f(a)$ の計算結果は常に同じとした．一方，不確かさの下で作動するプログラムあるいは不確かさを取り扱うプログラムの振舞いは確定的ではない．テスト実行毎に異なる結果を出力するかもしれない．テスト・オラクル $\mathcal{O}_T$ の考え方を拡張する必要がある．

2.2 統計的なオラクル

確率的な振舞いを示すプログラムは，入力データが同一の値であっても出力結果が実行毎に異なる可能性がある．そのような振舞いがプログラム内部の確率変数 p に依存することを $f_p(x)$ と表すことにする．ここで，確率変数 p が確率分布 ρ（$p \sim \rho$）にしたがうとした．

入力データ a を決めても，これに対する出力結果 y_a（$y_a = f_p(a)$）の値は一意に定まらない．素朴には，$p \sim \rho$ についての平均値を検査対象にする．入力を a としてプログラム実行 $f_p(a)$ を繰り返し，実行結果の集まりを $Y_a = \{y_a\}$ とする．Y_a の標本平均 μ_a を求めて統計的な推定を行えば良い [27]．

統計的なオラクル（Statistical Oracle）[14] は，仮説検定 [29] で，検査対象プログラムの欠陥有無を調べる方法である．正解の平均値 C_a が既知の時，帰無仮説を $\mu_a = C_a$，対立仮説を $\mu_a \neq C_a$ とする．プログラム実行の試行回数を N，${s_a}^2$ を不偏分散として，

$$t_a = \frac{\mu_a - C_a}{\sqrt{{s_a}^2/N}}$$

は，自由度 $N-1$ の t-分布にしたがう†1．両側検定を行って，$|t_a| > t_{\alpha/2}(N-1)$ の時，有意水準 α で帰無仮説を棄却する．つまり，$f_p(x)$ に欠陥があると推定する．

検査対象プログラム $g(x)$ が決定論的な振舞いをする場合でも，大規模な分散ソフトウェア・システム等では動作条件を一意的に定めることが難しいことが多い．入力に不確かさがあると考えて確率変数 a（$a \in P$）を導入する．$g \in P \to D'$ ならびに $a \sim \rho$ とすると，出力値 y の全体 $\{\ y \mid y = g(a) \wedge a \sim \rho\ \}$ が検査対象の挙動を表す．カオス・エンジニアリング [4] は，欠陥挿入した $g(x)$ を検査対象として，時系列をなす確率変数がもたらす安定状態の振舞い（Stable State Behavior）を異常検知の方法で調べる検査法である．

また，先の確率的な振舞いを示す検査対象 $f_p(x)$（$f_p \in D \to D'$）の確率変数 p を入力に明示して，$f \in D \to (P \to D')$ と表現しよう．入力値 $a \in D$ に対して，$f(a) \in P \to D'$ である．g を $f(a)$ に対応させれば，共に $P \to D'$ の形になる．多数回の $g(\_)$ を実行した結

†1 Y_a の母集団分布は正規分布とする．

果の平均 μ を求めて，また，この正解値 C が既知であれば，上記に示した仮説検定に基づく統計的なオラクルの適用が可能なことがわる．

2.3 統計的なメタモルフィック・テスティング

DNN ソフトウェアのようなテスト不可能プログラム [31] を検査する時は，ゴールデン出力を正解値の代替に使う部分オラクルの方法を用いる．このような検査対象 $f(x)$ が前項と同様に確率的な振舞いを示す場合，2 標本仮説検定の方法を応用する．検査対象を $f_{(1)}(x)$，ゴールデン出力を得るプログラムを $f_{(2)}(x)$ とし，各々の実行回数 N および M に対する統計指標（$j=1,2$ として，標本平均 $\mu_{(j)}$，不偏分散 ${s_{(j)}}^2$）を求める．

メタモルフィック・テスティングの方法 [5] [6] を用いる場合，検査対象を f，フォローアップ・テスト入力変換関数を T として，$f_{(1)}(a)=f(a)$ ならびに $f_{(2)}(a)=f(T(a))$ とする．検査対象プログラム $f(x)$ は共通なので，2 つの標本の母集団分布は同一の分散を持つとして良い．そこで，

$$s^2=\frac{(N-1){s_{(1)}}^2+(M-1){s_{(2)}}^2}{N+M-2}$$

として，

$$t_a=\frac{\mu_{(1)}-\mu_{(2)}}{\sqrt{s^2/N+s^2/M}}$$

を計算する．

前項の統計的なオラクルと同様に，両側検定を行って $|t_a|>t_{\alpha/2}(N+M-2)$ の時，有意水準 α で帰無仮説を棄却する．つまり，欠陥があると推定する．この方法を統計的なメタモルフィック・テスティング（SMT）と呼ぶ [11]．メタモルフィック関係 R_T として等価関係を用いる場合に対応する．

なお，検査対象 $f_{(1)}(x)$ に対して，$f_{(2)}(x)$ をデザイン多様性の方法で求めても良い．この時は，SMT の場合と異なり，2 つの母集団分布が同じ分散を持つと仮定できない．t_a 値の計算法を変更し，自由度にウェルチの方法（Welch Method）を用いる．

3 提案手法

部分オラクルと仮説検定の方法を一般化したテスティングのフレームワークを提案する．

3.1 フレームワーク

確率的な振舞いを示す検査対象プログラムを $\mathcal{F}_p{\in}V{\to}V'$，検査対象の値を求めるプログラムを $obs{\in}(V{\to}V'){\to}(V{\to}(D{\to}D'))$ とする．大きさ N の評価データの集まり $ES=\{\ e^{(j)}\ \}$ $(e^{(j)}{\in}D)$ に対して，$\mathcal{O}{\in}(V{\to}V'){\to}(V{\to}(\mathbb{P}D{\to}\mathbb{P}D'))$ を

$$\mathcal{O}(\mathcal{F}_p)(a)(ES)=\{\ (obs(\mathcal{F}_p)(a))(e^{(j)})\ \}$$

と定義する．$y^{(j)}{=}(obs(\mathcal{F}_p)(a))(e^{(j)})$ とすると，$\mathcal{O}(\mathcal{F}_p)(a)(ES)$ の平均値は

$$\mu_a\ =\ \frac{1}{N}\sum_{j=1}^{N}y^{(j)}$$

であり，また，不偏分散 ${s_a}^2$ も同様に

$${s_a}^2\ =\ \frac{1}{N-1}\sum_{j=1}^{N}(y^{(j)}-\mu_a)^2$$

と計算できる．

今，2 つのプログラムを $\mathcal{F}_{(1)}$ と $\mathcal{F}_{(2)}$ とし，共通する評価データ ES を準備する．$\mathcal{O}$ によって得られる標本の平均値ならびに不偏分散などを求めるプログラムを $\mathcal{U}$ とすると，$\mathcal{U}{\in}\mathbb{P}V{\to}V''$ である．ここで，得られる統計指標の型を V'' とした．たとえば，平均値，不偏分散，標本数の組とすると，$\mathcal{R}$ を実数，$\mathcal{N}$ を自然数として，$V''=\mathcal{R}{\times}\mathcal{R}{\times}\mathcal{N}$ である．

さらに，仮説検定に基づく推論機構を $\mathcal{H}_T$ と書き表すと，一般的なフレームワークを，次のように整理することができる．

$$\mathcal{H}_T(\ \mathcal{U}(\ \mathcal{O}(\underline{\mathcal{F}_{(1)}})(A_{(1)})(ES)\ ),\ \mathcal{U}(\ \mathcal{O}(\underline{\mathcal{F}_{(2)}})(A_{(2)})(ES)\ )\ )$$

素朴なオラクル $\mathcal{O}_T^{[N]}$ 等（第 2.1 節）が検査対象プログラムの入出力にのみ着目する一方，$\mathcal{H}_T$ は仮説検定の手続きにしたがった判定処理を含む．互いに部分オラクルとなる 2 つのプログラム $\mathcal{F}_{(1)}(A_{(1)})$ と $\mathcal{F}_{(2)}(A_{(2)})$ の統計的なオラクルの一般形になっている．また，評価用データ ES を $A_{(j)}$ から独立に導入可能とし，検査を柔軟に行えるようにした．具体的な方法は第 4 節の事例で説明する．

3.2 機械学習ソフトウェアの検査

深層ニューラル・ネットワークの教師あり学習タスクのソフトウェアについて，前記のフレーワークを具体化する．標準的な深層ニューラル・ネットワークの方法 [10] [12] では，訓練・学習プログラム $\mathcal{L}_f$ と予測・

推論プログラム $\mathcal{I}_f$ の2つが関わることに注意する．

学習モデルを $Y(W;x)$ とし，W は重み（学習パラメータ）の集まりを表す．$Y_W(x)$ と表記し，W をインデックスとする関数の集まりと考える．訓練済み学習モデルは具体的な値 W^c とする関数 $Y_{W^c}(x)$ である．$x{\in}V$，$Y_{W^c}{\in}V{\to}T$ とすると，学習モデルは，$Y{\in}W{\to}(V{\to}T)$ あるいは，インデックスを明示しないで，$Y{\in}\mathbb{P}(V{\to}T)$ である．

訓練データセット LS は多次元ベクトルと正解タグの組（データ点）の集まりなので $LS{\in}\mathbb{P}(V{\times}T)$ であり，$LS=\{\langle x^{(n)},t^{(n)}\rangle\}$ と表記する．特に，LS の経験分布を ρ^{EMP} とし，$\langle x^{(n)},t^{(n)}\rangle \sim \rho^{EMP}$ である．

訓練・学習プログラム $\mathcal{L}_f$ は，学習モデルを決め，訓練データセットを入力として，訓練済み学習モデル（関数）を求める（$\mathcal{L}_f(Y)(LS)$）．

$$\mathcal{L}_f{\in}\mathbb{P}(V{\to}T){\to}(\mathbb{P}(V{\times}T){\to}(V{\to}T))$$

と表せる．$\mathcal{L}_f$ は，具体的には，次の最適化問題を解いて得る関数 $Y(W^*;\_)$ を返すプログラムである．

$$W^* \;=\; \underset{W}{argmin}\; \mathcal{E}_{\langle x,t\rangle\sim\rho^{EMP}}[\![\, \ell(Y(W;x),t)\,]\!]$$

ここで，$\ell(\_,\_)$ を適切な損失関数とした．

予測・推論プログラム $\mathcal{I}_f$ は，訓練済み学習モデル（関数）を用いて，入力データのベクトルに対する予測・推論結果を返す（$\mathcal{I}_f(Y_{W^*})(\_)$）．

$$\mathcal{I}_f \in (V{\to}T){\to}(V{\to}V')$$

また，関数 $\mathcal{O}_f$ は，$\mathcal{I}_f$ と同様に，訓練済み学習モデル（関数）を用いる一方で，評価データの集まりに対する検査指標の標本を計算する（$\mathcal{O}_f(Y_{W^*})(\_)$）．$D$ と D' を適切に選んで，

$$\mathcal{O}_f \in (V{\to}T){\to}(\mathbb{P}\,D{\to}\mathbb{P}\,D')$$

である．評価データの集まり $ES=\{e^{(j)}\}$（$ES{\in}\mathbb{P}D$）とする．第 3.1 節と同様に，個々のデータに対して検査指標を計算する関数を obs とすると，$\mathcal{O}_f(Y_{W^*})(ES)=\{\; obs(Y_{W^*})(e^{(j)})\;\}$ である．この時，ES についての期待値は

$$\mu_{LS} \;=\; \frac{1}{N}\sum_{j=1}^{N} obs(\mathcal{L}_f(Y)(LS))(e^{(j)})$$

で，第 3.1 節の μ_a に対応する．

最後に，メタモルフィック・テスティングの場合，T をデータセット多様性に基づく変換関数 [17] [20] [21] として，$\mathcal{O}_f(\mathcal{L}_f(Y)(\underline{LS}))(\_)$ と $\mathcal{O}_f(\mathcal{L}_f(Y)(\underline{T(LS)}))(\_)$ を比較する．一方，学習モデルのデザイン多様性の場合，$\mathcal{O}_f(\mathcal{L}_f(\underline{Y_{(1)}})(LS))(\_)$ と $\mathcal{O}_f(\mathcal{L}_f(\underline{Y_{(2)}})(LS))(\_)$ を比較する．

3.3 検査指標の具体例

次に，検査指標を標本を求めるプログラム $\mathcal{O}_f$ の具体例 2 つを導入する．具体的な問題として，C 個のクラスへの分類タスクを考える．

3.3.1 予測の確からしさ

評価用データセットを $ES{\in}\mathbb{P}(V{\times}T)$ とする時，$V' \;=\; [0,1]^C$ として，予測・推論プログラム $\mathcal{I}_f(Y_{W^*})(x^{(j)})$ は入力データ $\langle x^{(j)},t^{(j)}\rangle{\in}ES$ のクラス $c{\in}[1,\cdots,C]$ に分類する確からしさを求める．この結果の値を C 次元ベクトル P（$P=\mathcal{I}_f(Y_{W^*})(x^{(j)})$）と表す．ここで，確からしさの値が最大となる分類カテゴリ（$c^*=argmax\,P[c]$）が，教師タグ $t^{(j)}$ の情報と一致する時，$\overset{c}{Y}_{W^*}$ は正解タグを再現する．正解タグがワンホット表現（One Hot）のベクトルであれば，$T=\{0,1\}^C$ で $t^{(j)}[c^*]=1$，$t^{(j)}[c]=0$（$c{\neq}c^*$）である．

予測の確からしさからなる標本は次のように求めることができる．

$$\mathcal{O}_{prob}(Y_{W^*})(ES)=\{\; P[c^*]\;\}$$

また，正解率は $t^{(j)}[c^*]=1$ となる頻度で，$\sharp S$ を有限集合 S の大きさとする時，$Acc(Y_{W^*})(ES) = \sharp\{j|t^{(j)}[c^*]=1\}/\sharp(ES)$ である．

3.3.2 内部活性率

次に，内部活性率を，Y_{W^*} の最後から 2 番目の層（Penultimate Layer）に対するニューロン・カバレッジ $\nu_{@PL}$ で定義する．

訓練済み学習モデル Y_{W^*} の構成要素のニューロン全体を $\{\nu_k\}$ とする．着目したニューロン ν_k の入出力関係を $o_k=\sigma_k(\sum_j w_j^*{\times}in_j)$ とし，閾値を th とする．入力が定まった時，出力が閾値より大きなニューロン（$o_k{>}th$）を活性状態にあるという．ニューロン・カバレッジ（NC）[26] は全ニューロンに対する活性状態ニューロンの比率として定義された．

今，$\nu_{@PL}{\in}(V{\to}T){\to}(V{\to}[0,1])$ として，$D \;=\; V$，$D' \;=\; [0,1]$ と選ぶ．$\nu_{@PL}(Y_{W^*})(x^{(j)}) \;=\; \sharp\{k|o_k{>}th\}/\sharp\{\nu_k\}$ を用いて，

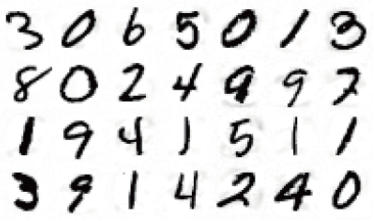


図 1　セマンティック・ノイズによる生成データ

$$\mathcal{O}_{act}(Y_{W^*})(ES) = \{\ \nu_{@PL}(Y_{W^*})(x^{(j)})\ \}$$

である．

4　事例

提案フレームワークでメタモルフィック・テスティングを用いる場合の検査実験結果を報告する．

4.1　訓練・学習ソフトウェアの検査

検査対象が満たすメタモルフィック性を考える．

4.1.1　MNIST 手書き数字分類タスク

手書き数字分類学習タスクの標準的なベンチマーク MNIST データセット†2 を用いて実験する．

MNIST は，手書き数字を 28×28 のピクセルからなる画像データを 784 次元ベクトルで表し，このベクトルと 0 から 9 の分類タグを組としてデータ点とする．各ピクセルはグレー階調を表し，0 から 255 の値をとるが，ベクトル表現ではピクセル値を 0.0 から 1.0 に規格化する．

60,000 個のデータ点を訓練データセット LS，10,000 個を試験データセット TS とする．LS と TS を合計 70,000 個のデータ・プールから均等に選ぶと，両者の経験分布に大きな差がない．訓練・学習では LS を入力し訓練済み学習モデルを求める．$Y_{W^*} = \mathcal{L}_f(Y)(LS)$ である．また，TS は LS と共に，訓練・学習過程で，正解率の計算に用いられ，汎化ギャップが生じるかを確認する．

4.1.2　フォローアップ・テスト入力生成

次に，テスト入力の生成方法を説明する．セマンティック・ノイズ生成関数を $\mathcal{S}_f$ とする．

$$\mathcal{S}_f \in (V \to T) \to (V \times T \to V)$$

ベクトルと正解タグの組 $\langle x, t\rangle$ と，訓練済み学習モデル Y_{W^*} が与えられた時，t を再現する新しいデータを $X^* = \mathcal{S}_f(Y_{W^*})(x,t)$ で求める．X^* は，入力データ x にセマンティック・ノイズが付加される．

ここで，$\mathcal{S}_f$ は L-BFGS [28] を基にした次の制約条件付き最適化問題 [22] を解くプログラムである．

$$X^* = \underset{X}{argmin}\ \ell(Y_{W^*}(X),\ t)\ +\ \lambda{\cdot}\ell'(X,\ x)$$
$$s.t.\ \ 0 \le X_j \le 1\ \ \wedge\ \ \Psi(X,x)$$

上式で $\ell(\_,\_)$ ならびに $\ell'(\_,\_)$ は損失関数で，たとえば，L_2 ノルムによって定義する．第 1 項は X の予測結果が正解タグ t を再現すること，第 2 項は X が与えられたデータ x に十分に似ていることを表す．ハイパー・パラメータ λ の値をうまく選ぶと，目視上，x と最適解 X^* の区別がつきにくくなる．

今，MNIST 訓練データセット LS を用いて訓練済み学習モデルを求めたとする．MNIST 試験データセット $LS = \{\langle x^{(n)}, t^{(n)}\rangle\}$ に，$\mathcal{S}_f$ を適用し，

$$LS' = \{\ \langle x^{*(n)}, t^{(n)}\rangle\ |\ x^{*(n)} = \mathcal{S}_f(\mathcal{L}_f(Y)(LS))(x^{(n)}, t^{(n)})\}$$

を得る．図 1 に画像の一部を示した．

次に，上記の LS から LS' を直接求める関数 $\mathcal{T}_x$ はデータセットの変換子

$$\mathcal{T}_x \in (V \to T) \to (\mathbb{P}(V \times T) \to \mathbb{P}(V \times T))$$

で，$LS' = \mathcal{T}_x(\mathcal{L}_f(Y)(LS))(LS)$ と表せる．簡単化して $\mathcal{T}_f(LS) \stackrel{def}{=} \mathcal{T}_x(\mathcal{L}_f(Y)(LS))(LS)$ とする．

$$\mathcal{T}_f \in \mathbb{P}(V \times T) \to \mathbb{P}(V \times T)$$

である．学習モデル $Y(W;\_)$ を決めれば，$\mathcal{T}_f$ は訓練データセット LS にのみ依存することがわかる．

$LS^{(0)} = LS$ とし，$LS^{(K)} = \mathcal{T}_f(LS^{(K-1)})$ $(K = 1, \cdots, N)$ とする．$LS^{(K)} = {\mathcal{T}_f}^K(LS)$ である．このようにして構成したデータセットの集まり $\{LS, LS^{(1)}, \cdots, LS^{(N)}\}$ はデータセット多様性 [20] [22] を示す．

4.1.3　メタモルフィック性

今，$Y_{W^{(K-1)*}} = \mathcal{L}_f(Y)(LS^{(K-1)})$ と表記する．$LS^{(K)}$ は $Y_{W^{(K-1)*}}$ をもとに最適化問題の解として生成されるので，$LS^{(K-1)}$ よりも $Y_{W^{(K-1)*}}$ に過適合す

†2　http://yann.lecun.com/exdb/mnist/

る．また，$Y_{W^{(K)*}} = \mathcal{L}_f(Y)(LS^{(K)})$ なので，$Y_{W^{(K)*}}$ は $LS^{(K)}$ に対する最適解であり，$Y_{W^{(K-1)*}}$ よりも $LS^{(K)}$ に過適合する．

繰り返し行うと，ある M に対して，$Y_{W^{(M)*}}$ と $Y_{W^{(M-1)*}}$ に有意な差が見られなくなり，$LS^{(M+1)}$ と $LS^{(M)}$ に差がなくなる．$LS^{(M+1)} \approx LS^{(M)}$ なので，$\mathcal{L}_f(Y)$ を適用しても $Y_{W^{(M)*}}$ が変わらない．その結果，M 以降，評価用データセット ES に対する内部活性率が一定になる．

訓練・学習プログラム $\mathcal{L}_f(Y)$ に欠陥がなければ，LS を初期値として $\mathcal{T}_f$ を繰り返し適用し，M 回目と $M+1$ 回目の内部活性率が一致する．繰り返し適用しても，一致しなければ欠陥があると推測できる．内部活性率は評価データ ES についての $\nu_{@PL}$ の平均値なので，一致しないことを仮説検定の方法で確認すれば良い．

4.2 対照実験

セマンティック・ノイズを利用したメタモルフィック性の方法では，$\mathcal{T}_f$ の繰り返し適用回数を実験的に調べる必要がある．最適化問題（第 4.1.2 節）の形から，繰り返し回数 M は大きな値にならないと考えられる．対照実験によって確認しよう．

標準的な学習方式を実現したプログラム $\mathcal{L}_f^{PC}$ を準備し，これに欠陥挿入したプログラム $\mathcal{L}_f^{BI}$ を作成した [21] [22]．PC は Probably Correct（おそらく正しい），BI は Bug Injected（バグ挿入した）を示す．また，学習モデル $Y_W(\_)$ は PC と BI で共通で，古典的な全結合ニューラル・ネットワーク（Fully Connected）とする．さらに，活性関数を $ReLU$ とし，$\nu_{@PL}$ 計算に用いる閾値を 0 とする．

2 つの訓練・学習プログラムを対象として，次の手順で実験する．ここで，記号 MD は PC あるいは BI を示し，PC の実験系列と BI の系列を並行して実施する．また，MNIST のデータセット LS と TS を各々 $LS^{(0)}{}_{MD}$（$TS^{(0)}{}_{MD}$）とする．$LS = LS^{(0)}{}_{PC} = LS^{(0)}{}_{BI}$ （$TS = TS^{(0)}{}_{PC} = TS^{(0)}{}_{BI}$）である．一方，$K \geq 1$ の時，$LS^{(K)}{}_{PC}$ と $LS^{(K)}{}_{BI}$ （$TS^{(K)}{}_{PC}$ と $TS^{(K)}{}_{BI}$）は一致しない．$K = 0$ から開始する．

1. $LS^{(K)}{}_{MD}$ と $TS^{(K)}{}_{MD}$ を用いて訓練・学習を行う．$Y^{MD}_{W^{(K)*}} = \mathcal{L}_f^{MD}(Y)(LS^{(K)}{}_{MD})$
2. TS を評価データセットとして正解率と内部活性率を測定する．$Acc(Y^{MD}_{W^{(K)*}})(TS)$ と $\mathcal{O}_{act}(Y^{MD}_{W^{(K)*}})(TS)$ である．
3. $Y^{MD}_{W^{(K)*}}$ を利用して，セマンティック・ノイズを挿入したデータセット $LS^{(K+1)}{}_{MD}$ と $TS^{(K+1)}{}_{MD}$ を作成する．$LS^{(K+1)}{}_{MD} = \mathcal{T}_f(LS^{(K)}{}_{MD})$ と $TS^{(K+1)}{}_{MD} = \mathcal{T}_f(TS^{(K)}{}_{MD})$ である．
4. $K = K + 1$ として繰り返す．

また，$K \geq 1$ の時，内部活性率（上記ステップ 2）に対する仮説検定の方法によって，メタモルフィック関係（第 4.1.3 節）を調べる．つまり，

$$H_T^{MD}(\ \mathcal{U}(\ \mathcal{O}_{act}(\mathcal{L}_f^{MD}(LS^{(K-1)}))(TS)\ ),\ \mathcal{U}(\ \mathcal{O}_{act}(\mathcal{L}_f^{MD}(LS^{(K)}))(TS)\ )\ )$$

である．同じ評価用データを用いることから，2 つの標本数は同一（$N = M$）である．この時，t-値は簡単になり，

$$t = \frac{\mu_{(1)} - \mu_{(2)}}{\sqrt{({s_{(1)}}^2 + {s_{(2)}}^2)/N}}$$

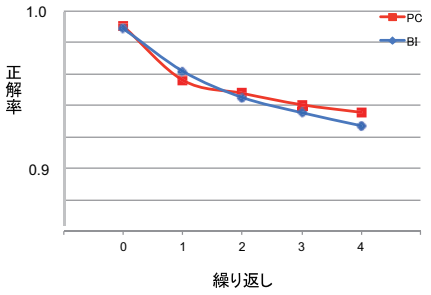
を計算すれば良い．また，N は十分に大きいので，自由度 ∞ の t-分布にしたがうとする．

4.3 実験結果と考察

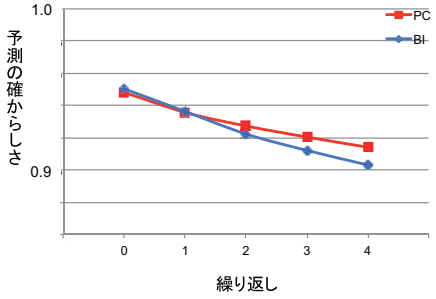
図 2 は横軸に繰り返し適用の回数 K（${\mathcal{T}_f}^K$），縦軸に正解率 Acc（図 2(a)）、予測の確からしさ $\mathcal{O}_{prob}$（図 2(b)），および内部活性率 $\mathcal{O}_{act}$（図 2(c)）をプロットした．

正解率は，PC と BI 共に 90% を超える値をとり，その違いが現れにくい．つまり，正解率を監視していても欠陥の有無が判断しにくい．また，繰り返しと共に単調減少するのは，各ステップで用いた訓練データセット $LS^{(K)}$ の経験分布と評価に用いたデータセット TS の経験分布の違い（データ・シフト）が大きくなるからである．さらに，$K \geq 2$ では，BI の予測の確からしさが PC よりも劣る．データ・シフトが大きくなると共に確からしさが低下する．つまり，欠陥があると，ロバスト性に影響する．

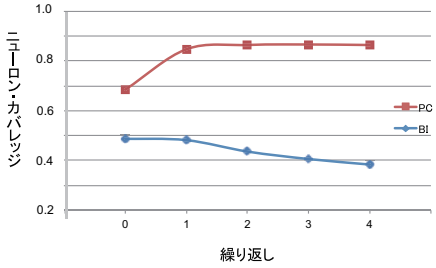
内部活性率の値は PC と BI で大きく異なる．PC が 60% 以上のニューロンを有効に使用しているのに



(a) 正解率



(b) 予測の確からしさ



(c) 内部活性率

図 2　指標の変化

対して，BI は 40% 程度のニューロンだけが予測・推論に関わる．また，繰り返しと共に，単調増加する PC に比べて，BI は単調減少する．BI がモデル・キャパシティを有効に活用していないことを示唆する．このように，有効利用できていないことが，$K \geq 2$ でロバスト性の段階的な低下として現れる原因と考えられる．

図 3 は縦軸に内部活性率についての t-値を示す．横軸は仮説検定の対象であって，$\mathcal{T}_f{}^K$ と $\mathcal{T}_f{}^{K+1}$ の比較を表す．点線は，有意水準 0.001/2 (0.0005) の t-値に対応する．両側検定によると，グラフから BI は有意水準 0.001 以下で一致しないと推測でき，BI に欠

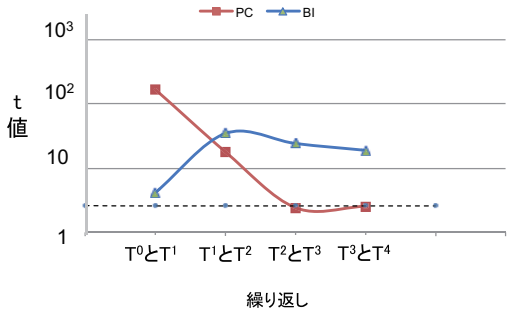


図 3　t-値の変化

陥挿入したことと整合している．

なお，仮説検定の方法を用いたので，$M \geq 2$ で PC が収束する，つまり，欠陥がないと結論できるわけでないことに注意して欲しい．通常のソフトウェア・テスティングと同様に，欠陥があることを確認する方法である．

5　議論

5 つの観点から既存研究との関係を整理する．

統計的な検査法　統計的な検査の方法は，第 2 節に触れたように，確率的な挙動を示すプログラムのテスティングに用いられている [14] [11] [27]．一方，CPS [18] の代表的な計算モデルとして用いられるハイブリッド・システムを対象とした形式検証に確率・統計的な方法を用いる研究 がある．統計モデル検査 [32] と呼ばれるが，実行時検証と仮説検定を組み合わせた手法である．このような統計モデル検査と統計的なソフトウェア・テスティングには共通点があり，統一的な視点から整理可能である．

本稿のフレームワークは，部分オラクルと仮説検定を組み合わせた統計的なメタモルフィック・テスティング [11] を再考し，テスト・オラクルの定義（第 3.1 節）を関数モデリングの方法で明確化した．

内部活性　ニューロン・カバレッジ（NC）は，訓練データ補完の指針を得る方法として提案された [26]．NC 値が小さい場合，訓練データセットに問題があるとする．NC を増加するようなデータを自動合成し，訓練データセットに追加する．

本稿では，訓練済み学習モデルの特定の層の NC 値

をもとに内部活性率を定義した．検査対象の訓練・学習プログラムが欠陥を含む場合，欠陥がないプログラムに比べて，内部活性率が小さくなることを実験で確認した．内部活性率に着目することで，訓練データセットの不足あるいは偏り，訓練・学習プログラムの欠陥，といった何らかの問題があることが従来論じられていた [23]．この考察結果をもとにメタモルフィック関係を定義した．特に，仮説検定の方法を組み合わせた点が新しい．

セマンティック・ノイズ 実験では，セマンティック・ノイズ [22] をもとにしたフォローップ・テスト入力生成と組み合わせることで，内部活性率をメタモルフィック関係から参照する検査法を用いた．このセマンティック・ノイズ生成関数 S_f によって得られるデータは，この関数が利用する訓練済み学習モデルを得る際に用いたデータセット LS の経験分布との関係が明確でない．この点，基本となるデータ分布との関係性が明確な GAN [9] と異なることに注意して欲しい．テスト入力作成の観点では，GAN による方法が想定する条件（LS の経験分布）下での網羅性向上であるのに対して，本稿の S_f が生成するデータは，基本とした LS からみて大きな偏りを持つ可能性のあるファズ [15] といえる．

T の繰り返し適用 実験では，フォローアップ・テスト入力生成関数 T を繰り返し適用し，生成した入力データに対する実行結果をメタモルフィック関係の評価対象とした．この時，さまざまなデータセットが生成されることを，データセット多様性 [20] と呼ぶ．このデータセット多様性による繰り返し法を機械学習ソフトウェアの検査に適用するアイデアは SVM の検査に見られる [17]．また，ニューラル・ネットワーク訓練・学習プログラムの検査への応用 [19] が論じられ，ネットワーク構造の情報を利用したホワイトボックス的な方法が試みられた [21]．

本稿では，より適用範囲を広げることを目的として，最適化問題の定式化によって，ブラックボックス的にテスト入力を自動生成する方法を導入し，統計的なオラクルを用いるテスティングの実験を行った．

評価用データ分布 統計的な動的検査では，多様な状況を得る方法として，ランダム生成したデータを用いる．一方で，ランダム生成の方法では，欠陥箇所を実行する頻度が小さくなる．そこで，生成データ分布に偏りを持たせてコーナーケース・テスティングを行い，検査結果の統計的な信頼度を改善する方法が知られている．たとえば，モンテカルロ法で知られている重点サンプリングの方法を用いた事例 [16] や深層ニューラル・ネットワークのロバスト性検査に用いられている [30]．

本稿が検査対象とする訓練・学習プログラムでは，検査に用いる評価用データセットと欠陥箇所の関係が明らかでないことから，コーナーケースを重点的に調べるような偏りを導入することが難しい．逆に，検査対象問題の性質を忠実に反映する評価用データを用いるほうが好ましい．そこで，試験データセットを評価用に用いた．$LS^{(K)}$ と TS は同じ手書き数字分類の学習タスクを適切に表す一方，データ・シフトが生じている．訓練・学習に用いられる $LS^{(K)}$ を基準とすると，TS は適切に生成されたファズ [15] と同様な役割を担う．

6 おわりに

統計的なオラクルと部分オラクルを組み合わせた一般的なテスティング・フレームワークを整理し，ニューラル・ネットワークの訓練・学習プログラムのデバッグ・テスティングへの適用例として簡単な分類学習の MNIST データセットを用いた実験を報告した．

機械学習ソフトウェア開発では，オープン・ソフトウェアとして公開されている既存フレームワークを使うことが多い．本稿の方法は，訓練・学習プログラム $\mathcal{L}_f$ を対象とするものであり，既存フレームワークの品質確認に用いることができるという点で，他の機械学習テスティングの研究 [33] と異なる．深層ニューラル・ネットワークのテスティング技術として新しい視点を導入したといえる．今後，他のデータセットへの適用を通して，提案方法の実用的な有用性を確認していきたい．

謝辞 本研究の一部は JSPS 科研費 JP18H03224 の助成および NEDO の委託業務（NII へは再委託）によって実施した．実験に協力して下さった今井克則氏（アイティスミス）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*, 37 (4), 1988, pp.418-425.

[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), 2015, pp.507-525.

[4] Basiri, A., Behnam, N., de Rooji, Hochstein, R.L., Kosewski, L., Reynolds, J.,and Rosenthal, C.: Chaos Engineering, *IEEE Software*, 33 (3), 2016, pp.35-41.

[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, and also arXiv:2002.12543, 2020.

[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, 2018, pp. 1-27.

[7] Davies, M. and Weyuker, E. : Pseudo-oracles for Non-testable Programs, In *Proc. ACM'81 Conference*, 1981, pp.254-257.

[8] Elbaum, S. and Rosenblum, D.S.: Known Unknowns - Testing in the Presence of Uncertainty, In *Proc. 22nd FSE*, 2014, pp.833-836.

[9] 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 27*, 2014, pp.2672-2680.

[10] Goodfellow, I., Bengio, Y., and Courville, A.: *Deep Learning*, The MIT Press 2016. [邦訳] 岩澤有祐，鈴木雅大，中山浩太郎，松尾豊 監修，味曽野雅史，黒滝紘生，保住純，野中尚輝，河野慎，冨山翔司，角田貴大 共訳，深層学習，KADOKAWA 2018.

[11] Guderlei, R., and Mayer, J.: Statistical Metamorphic Testing - Testing Programs with Random Output by Means of Statistical Hypothesis Tests and Metamorphic Testing, In *Proc. 7th QSIC*, 2007, pp.404-409.

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

[13] Howden, W.E.: Theoretical and Empirical Studies of Program Testing, In *Proc. 3rd ICSE*, 1978, pp.305-311.

[14] Mayer, J. and Guderlei, R.: Test Oracles Using Statistical Methods, In *Proc. 1st SOQUA*, 2004, pp.179-189.

[15] Miller, B.P. , Fredricksen, L. , and So, B. : An Empirical Study of the Reliability of UNIX Utilities, *Comm. ACM*, 33 (12), 1990, pp.32-44.

[16] 中島震: 実行時干渉の発生確率予測，コンピュータ・ソフトウェア, 30 (3), 2013, pp.95-101.

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

[18] 中島震：CPS:そのビジョンとテクノロジー，研究/技術/計画，32 (3), 2017, pp.235-250.

[19] Nakajima, S.: Generalized Oracle for Testing Machine Learning Computer Programs, In *Proc. SEFM Workshop*, 2017, pp.174-179.

[20] 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), 2018, pp.26-32.

[21] Nakajima, S. : Dataset Diversity for Metamorphic Testing of Machine Learning Software, In *Proc. 8th SOFL+MSVL*, 2019, pp.21-38.

[22] Nakajima, S. and Chen, T.Y.: Generating Biased Dataset for Metamorphic Testing of Machine Learning Programs, In *Proc. 31st IFIP-ICTSS*, 2019, pp.56-64.

[23] Nakajima, S.: Distortion and Faults in Machine Learning Software, *Proc. 9th SOFL+MSVL*, 2020, pp.29-41, and also arXiv:1911.11596, 2019.

[24] 中谷多哉子，中島震: ソフトウェア工学，放送大学教育振興会 2019.

[25] Ng, A. and Soo, K.: Numsense! Data Science for the Layman: No Math Added, Brite Koncept Ltd., 2017. [邦訳] 上藤一郎 訳: 数式なしでわかるデータサイエンス，オーム社 2019.

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

[27] Servcikova, H., Borning, A., Socha, D., and Bleek, W.-G.: Automated Testing of Stochastic Systems: A Statistically Grounded Approach, In *Proc. ISSTA'06*, 2006, pp.215-224.

[28] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R.: Intriguing properties of neural networks, In *Proc. ICLR*, 2014, and also arXiv:1312.6199, 2013.

[29] 東京大学教養学部統計学教室（編）: 統計学入門，東京大学出版 1991.

[30] Webb, S., Rainforth, T., Teh, Y.W., and Kumar, M.P.: A Statistical Approach to Assessing Neural Network Robustness, In *Proc. ICLR*, 2019, and also arXiv:1811.07209, 2018.

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

[32] Younes, H.L.S., Kwiatkowska, M., Norman, G.and Parker, D. : Numerical vs. Statistical Probabilistic Model Checking, *J. STTT*, 8 (3), 2006, pp. 216-228.

[33] Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine Learning Testing: Survey, Landscapes and Horizons, arXiv:1906.10742, 2019.