Vol.2017-SE-197 No.9
2017/11/10

ニューラル・ネットワークのメタモルフィック・テスティング

中島 震[1,a)]

概要：統計的な機械学習ソフトウェアは計算結果が既知でないことから，データ多様性に着目した疑似オラクルを用いるメタモルフィック・テスティングが有効な方法である．ところが，ニューラル・ネットワークは非凸最適化問題であり，従来の方法を適用することが難しい．本稿では，データセット・カバレッジを一般化したデータセット多様性と，最適化の繰り返し処理過程で得られる時系列を調べる振舞いオラクルという 2 つの考え方を導入した新しいメタモルフィック・テスティング法を提案する．手書き数字認識学習タスクの実験を通して提案方法の効果について考察する．

キーワード：機械学習プログラム，疑似オラクル，データセット多様性，振舞いオラクル

Metamorphic Testing of Neural Networks

Shin Nakajima[1,a)]

Abstract: Because computation results of machine learning programs are not known in advance, their software testing often adapts the metamorphic testing method which uses pseudo oracles based on data diversity. The learning problem of neural networks is non-convex optimization, and thus the approaches with data diversity are inadequate. This paper proposes a new metamorphic testing method to be based on dataset diversity and behavioral oracle. It also presents some lessons learned from experiments of applying the method to recognizing handwritten numbers.

Keywords: Machine Learning Programs, Pseudo Oracle, Dataset Diversity, Behavioral Oracle

1. はじめに

統計的な機械学習の研究が進み，画像認識をはじめとして，実用的な機能ならびに性能を持つ機械学習ソフトウェアが登場した．今後，自動運転などに組込まれることが期待されているが，そのようなシステムは不具合の影響が大きいことから，ディペンダビリティを高める技術の確立が急務になっている．特に，学習フェーズを実現したプログラムに不具合がないことを確認したい．ところが，学習結果を予め知ることができないことから，正解値を予測することが困難なプログラムである．

機械学習ソフトウェアは，計算結果が予測できないという点で，従来のソフトウェア・テスティングの技術 [2] を適用することができない．正解値が未知のプログラムを対象とする疑似オラクルの方法 [13] を用いる．データ多様性 [1] に着目したメタモルフィック・テスティングの方法 [4] が，サポート・ベクタ・マシン等の教師あり分類学習タスクのソフトウェア・テスティングで成功を収めた [14]．データセット中のデータ分布をうまく作り出すデータセット多様性に着目することで，さらに効率よくテスティングを行えることがわかった [8][9]．

統計的な機械学習の古典ともいえるニューラル・ネットワーク [3] は，非凸最適化問題であり，一般に大域的な最適地点に到るかが自明ではない．妥当と思われる収束点に達することを繰り返し処理の中で監視する．疑似オラクルの

[1] 国立情報学研究所
NII, Chiyoda-Ku, Tokyo 101—8430, Japan
[a)] nkjm@nii.ac.jp

ⓒ 2017 Information Processing Society of Japan

表 1　一般的なメタモルフィック性

Additive	データ点の属性に加算
Multiplicative	データ点の属性に乗算
Permutative	データ点の入れ替え
Invertive	正解ラベルの反転
Inclusive	新しいデータ点の追加
Exclusive	既存データ点の削除

方法を用いる場合であっても，収束時のスナップショットだけで評価することはできない [10]．本稿では，求める学習パラメータから得られる統計値の時系列を調べる振舞いオラクル [11] の考え方を導入する．そして，データセット多様性 [12] と振舞いオラクルを組み合わせた新しいメタモルフィック・テスティングの方法を提案する．手書き数字認識学習タスクの実験を行って提案方法の有効性を示す．

2. メタモルフィック・テスティング

入力 x に対する正解値 C^x が事前に知られていない場合を考える．一般に，欲しい値が未知のプログラムは正解値がわからない．知っていればプログラムを作る必要がない．このようなプログラムの検査では，相対的な正解値を C^x の代わりに使う疑似オラクルを用いる [13]．

相対的な正解値は，検査対象以外の他プログラムが計算した値（黄金出力）である．プログラムは正しさが保証されていないので，その計算値は絶対的に正しいとはいえない．相対的な正解値を得る方法として，デザイン多様性とデータ多様性の 2 つの考え方がある [1]．デザイン多様性に基づく方法は，同じ機能仕様を満たすプログラムを複数開発するもので，N バージョン・プログラミングがある．独立した開発チームが異なるプログラミング言語を用いる等の工夫によって，疑似オラクルの信用レベルを向上させる．

データ多様性は，数値計算など多様なデータによる検査が必要なプログラムで有用な考え方で，メタモルフィック・テスティング [4] がある．入力 $x^{(m)}$ に対する計算結果を $f(x^{(m)})$ とする．$x^{(m)}$ に変換 T を施してフォローアップ入力 $x^{(m+1)} = T(x^{(m)})$ を得る．ここで，変換 T は 2 つの計算結果の間に適切なメタモルフィック関係 Rel^T が成り立つように，検査対象プログラムの機能仕様から見出す．2 つの計算結果が，$Rel^T(f(X^{(m)}),\ f(X^{(m+1)}))$ を満たさない時，f に不具合があると結論する．

メタモルフィック・テスティングは，サポート・ベクタ・マシン（SVM）等の教師あり分類学習タスクで成功を収めた．文献 [14] は，フォローアップ入力が満たすメタモルフィック性の分類し，表 1 として整理した．さまざまな学習タスクに応用可能な一般的な性質を表している．

3. 提案方法

3.1 教師あり機械学習

一般に，教師あり機械学習の基本的な考え方は，最適化の問題として理解することができる．データセット D をデータ点 $\langle \vec{x}^n, y^n \rangle$ の集まりとする．ここで，$\vec{x}^n$ は多次元ベクタ，y^n は教師ラベルを表す．

統計的な機械学習は，パラメータ θ で特徴つけられた関数 $F^C(\theta;\ \vec{x})$ の集まり $\{F^C(\theta;\ \vec{x})\}^\theta$ から，与えられたデータセット D に対する目的関数 $\mathcal{E}(\theta;\ \{\langle \vec{x}^n, y^n \rangle\})$ を最小にするパラメータ θ^* を求めることである[*1]．つまり，$\theta^* = arg\,min\ \ \mathcal{E}(\theta;\ \{\langle \vec{x}^n, y^n \rangle\})$ である．学習問題を具体的に決めるハイパーパラメータ C が存在することが多く，関数 $F^C(\theta^*;\ \vec{x})$ を確定するには，C の値を決めておく必要がある．最適化問題からみると，C は定数であり，θ^* は C に依存する（$\theta^*(C)$）．

パラメータ推定に用いるデータセットを訓練データセット D_R，得られた関数 $F^C(\theta^*;\ \vec{x})$ の評価に用いるものを試験データセット D_T と呼ぶ．何らかの方法で決めたハイパーパラメータ値を C_o とする時，推定したパラメータ $\theta^*(C_o)$ の良し悪しは，試験データセット D_T に関数 F^{C_o} を適用した結果として得る正解率で評価する．

3.2 データセット多様性

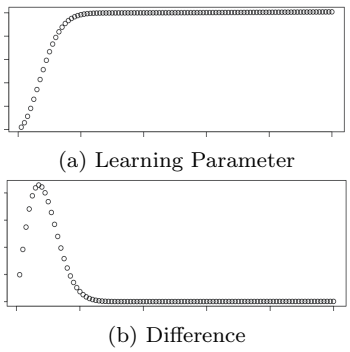
検査対象 f は目的関数 $\mathcal{E}$ を最小とする最適化問題を解く数値計算プログラムで，与えられた入力データセット D_R の要素全体に対する処理として実現される．単純な制御構造を持つ繰り返し処理であることから，制御グラフを対象とする通常のテスト・カバレッジは自ずから満たされ良い基準にならない [8]．

一方，f の検査結果は入力データセット D_R に依存する．より詳しくは，D_R 中のデータ点が，学習タスク毎に特徴的な分布をなす場合，計算結果に大きな影響を与えることがある．一般に，異なるデータセットは無限にあるので，分布をランダムに生成しても，期待する特徴を持つデータセットを得られる保証はない．対象学習タスクの性質に依存して適切な偏りを持つ分布を得たい．文献 [9] では，SVM を対象としたデータセット・カバレッジを導入した．関数 f を検査するという観点から，入力データセット D_R 内のデータ分布を工夫し，限界データ点を系統的に導入する方法を示した．表 1 の Inclusive 性の一例になっている．

本稿では，文献 [12] で論じたデータセット多様性を考慮したメタモルフィック・テスティングを採用する．これは，データセット・カバレッジ [9] の一般化である．

テスティングの入力として用いる訓練データセットを $D^{(m)}$ と表記する．目的とする学習対象の訓練データセッ

[*1] 制約条件 $\mathcal{C}(\theta;\ \{\langle \vec{x}^n, y^n \rangle\})$ を伴うこともある．



(a) Learning Parameter

(b) Difference

図 1 Shape of Indicator Graph

トを $D^{(0)}$ とする．データセット多様性では，フォローアップ入力を得る変換関数 T を拡張し，$T(D^{(m)},\ f(D^{(m)}))$ とする．フォローアップ入力 $D^{(m+1)}$ を得るのに，計算結果 $f(D^{(m)})$ を用いることを示す．つまり，m 回目に得られた計算結果を利用して，$m+1$ 回目のフォローアップ入力を生成する．文献 [9] で用いた Inclusive 性は，$D^{(m)}$ に対して得られた分離境界線の近傍に新しいデータ点を分布させることだった．たしかに，計算結果 $f(D^{(m)})$ を使っている．

3.3 振舞いオラクル

SVM のような凸最適化問題では，原理的には，探索初期値に依存しないで，大域的な最小解に収束することが保証されている．収束と判定された時点での学習パラメータがメタモルフィック関係を満たすか否かを検査すれば良い．SVM の事例では，得られたラグランジュ乗数によって定義される分離超平面式の関係を調べた．

NN のような非凸最適化問題では，求める学習パラメータの初期値が，収束解に影響を与える．初期値を適切に選ぶと大域的な最小解に達する一方，初期値の選択を誤ると局所的な極小解に陥る．そこで，解探索の繰り返し単位 (エポック) ごとに，プログラムの振舞いを要約した指標の変化を監視する．つまり，指標の時系列を調べる．

一般に学習パラメータの数は膨大なので，その全てを指標にする検査は困難である．SVM の場合は分離超平面を定義する 2 つのパラメータが良い要約情報だった．以下では，学習パラメータから計算可能な何らかの統計量の時系列を考える．

図 1(a) は，エポック進行に対する学習パラメータ値の定性的な変化を模式的に表す．仮に初期値を 0 とすると，学習の進行と共に，有意な値をとることから例えば値が増加し，その後，一定となって収束する．学習の進行具合は，図 1(b) にような変化から作られる指標グラフを調べるとわかりやすい．つまり，統計量を α とする時，指標グラフ $g[\alpha](e)$ を次のように定義する．

$$g[\alpha](e) = \alpha(e+1) - \alpha(e)$$

指標グラフ $g[\alpha](e)$ は統計量 α の変化の傾向を表す．以下の正規化に関する説明にあるように，大きく変化するエポックを知ることを目的として導入した．

次に，2 つのグラフの近さを表す距離 $\mathcal{F}(g_1, g_2)$ を定義する．ただし，2 つのグラフは正規化済みとする．今，エポック数を E とすると，

$$\mathcal{F}(g_1, g_2) = \tfrac{1}{E}\textstyle\sum_e |g_1(e) - g_2(e)|^2$$

となる．ある微少量 ϵ に対して，$\mathcal{F}(g_1, g_2) \leq \epsilon$ が成り立つ時，2 つのグラフは一致と判断する．本稿では，このようなグラフ間の距離を検査の基準とする方法を振舞いオラクルと呼ぶ．

ここで，指標グラフの形を，どのような傾向を示すかを調べる．第 1 に，メタモルフィック・テスティングの基本的な考え方は，$f(D^{(m)})$ と $f(D^{(m+1)})$ を比較することだった．フォローアップ入力のデータセット $D^{(m+1)}$ はオリジナルのデータセット $D^{(m)}$ と，同一学習タスクの対象であるという点で強い関係がある．理想的には，同じ母分布の異なる標本と考えられよう．そこで，学習プログラムが方式通りに構築されていて欠陥がなければ，求める学習パラメータを統計量とする指標グラフは，ほぼ同じ形を示すと期待できる．つまり，同じ母分布の異なる標本に対する振舞いが大きく違えば，その学習方式そのものが良いとはいえない．特定の訓練データセットに過適合するような学習方式の恐れを排除できない．本稿で想定する学習方式は，このような不具合はないと仮定する．

第 2 に，学習方式と学習プログラムに不具合がない場合であっても，指標グラフの形は，ハイパーパラメータの違いに影響を受ける．ハイパーパラメータ値が適切でない時，図 1(a) の収束はなだらかなる．つまり，図 1(b) のピークが右にずれ尖度が緩くなる．単純に指標グラフを比較すると，ハイパーパラメータの違いが，距離の判定に影響を与える．そこで，このような指標グラフを相似形と見做して同一視するように，距離計算の前処理として正規化を施す方法を採用する．

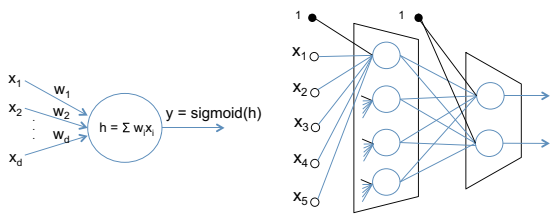
正規化は，比較する 2 つのグラフのピーク位置のエポックが一致するように，横軸を圧縮あるいは伸張することで行う．また，縦軸は，最小値が 0，最大値が 1 になるようにスケールを揃えれば良い．統計量の絶対値よりも，グラフの全体的な形を知りたいのである．

以上の準備を行って，メタモルフィック関係を距離によって定義する．つまり，関係を

$$\begin{aligned} &Rel^T(f(D^{(m)}),\ f(D^{(m+1)})) \\ &= \textstyle\bigwedge_\alpha \left(\mathcal{F}(g[\alpha^{(m)}],\ g[\alpha^{(m+1)}]) \leq \epsilon \right) \end{aligned}$$

とする．学習パラメータから得られる統計量 α を，学習タスクや方式ごとに適切に選べば良い．

ⓒ 2017 Information Processing Society of Japan



(a) Perceptron (b) Two Layered Network

図 2 Neural Network

4. ニューラル・ネットワーク

4.1 非凸最適化問題

ニューラル・ネットは，回帰や分類（識別）などの学習タスクに利用可能な一般的な枠組みである（たとえば，[3] の第 5 章）.

図 2 (a) はニューラル・ネットの基本構成になるパセプロトンの模式図．一連の入力信号 x_i を受け取り，$y = \sigma(\sum_{i=1}^{d} w_i x_i)$ の計算結果を出力する．ここで，σ は活性化関数と呼ばれる非線形関数である．

本稿で考察する古典的なニューラル・ネットは図 2 (b) のように，パセプトロンを 2 層に重ねた構造を持つ．D 個の入力信号は M 個の中間層（隠れ層）に伝播され，その後，R 個の出力層につながる．今，h と r を活性化関数とする時，k 番目の出力信号 y_k は次のように表せる．

$$y_k(\mathbf{V}, \mathbf{W};\ \vec{x}) = r(\ \sum_{j=0}^{M}\ v_{kj} h(\sum_{i=0}^{D}\ w_{ji} x_i)\ )$$

ここで，$\vec{x}$ は D 次元ベクタ（$D\times1$ 行列）である．$\mathbf{W}$ は隠れ層への入力リンクの重み w_{ji} を要素とする $M\times D$ 行列，$\mathbf{V}$ は出力層への入力リンクの重み v_{kj} を要素とする $R\times M$ 行列，とする．行列の乗算を $\cdot$ で明示すると，$\vec{y}$ を R 次元の出力信号として，

$$\vec{y}(\mathbf{V}, \mathbf{W};\ \vec{x}) = r(\mathbf{V}\cdot h(\mathbf{W}\vec{x}))$$

と書ける．今，正解ラベルを $\vec{t}^n$ として，N 個の入力データ点 $\{\langle \vec{x}^n,\ \vec{t}^n \rangle\}$ $(n = 1, \ldots,\ N)$, からなるデータセットが与えられたとする．誤差二乗和で定義する損失関数

$$E(\mathbf{V}, \mathbf{W};\ \{\vec{t}^n, \vec{x}^n\}) = \tfrac{1}{2}\sum \left\| \vec{t}^n - \vec{y}(\mathbf{V}, \mathbf{W}; \vec{x}^n) \right\|^2$$

を使って，

$$arg\min_{\mathbf{V},\mathbf{W}}\ E(\mathbf{V}, \mathbf{W};\ \{\vec{t}^n, \vec{x}^n\})$$

と定式化できる．

4.2 時系列

損失関数を最小化する非凸最適化問題の基本解法は数値的に解く勾配法である．学習率と呼ぶハイパーパラメータ η $(>\ 0)$ を与えて，$\nabla E(\mathbf{V}^{old}, \mathbf{W}^{old})$ から更新値を繰り返し求める．実際は，処理性能向上の工夫として，逆伝播法や確率勾配法などを用いる．

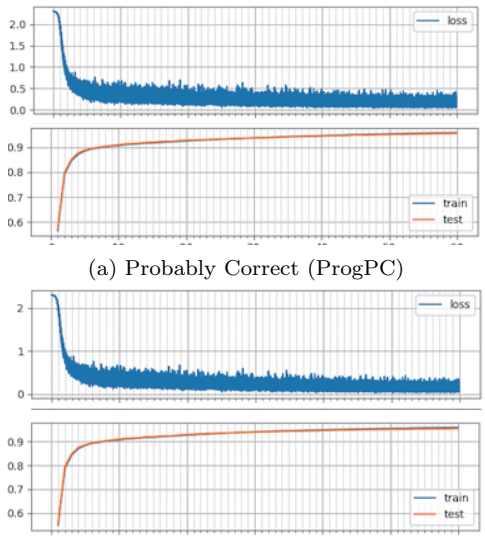
repeat {
$\quad \mathbf{V}^{new} = \mathbf{V}^{old} - \eta\nabla_v E;$
$\quad \mathbf{W}^{new} = \mathbf{W}^{old} - \eta\nabla_w E$
} **until** $(converge \vee timeout)$

訓練データセットの要素であるデータ点は多次元ベクターである．その各属性値にバラツキがあると，値が大きい属性の影響が強く出てしまう．そこで，入力正規化を前処理として施す．たとえば，$\vec{x}^n$ のすべての属性 x_i^n を属性 i ごとに，正規分布 $Norm(0, 1)$ に変換する方法，範囲 $[0,\ 1]$ とする方法などがある．

先に述べたように，非凸最適化問題であることから，重み値の初期値に依存して収束解が異なることがある．重み値の初期値を適切に選ぶと，大局的な最小解に収束するが，初期値の選択に誤ると局所解に陥る．そこで，重み初期値の選び方が重要なヒューリスティックスとなる．たとえば，std^2 として，定数 0.01 や $2/N$（N はソース・ノードの数）を選び，重みの全体が正規分布 $Norm(0, std^2)$ にしたがうようにする方法がある．

一般に最小解への収束が保証されていないので，繰り返しの単位（エポック）ごとに，損失関数を調べて収束に到る傾向を監視する．また，妥当な解に収束していることを確認する方法として，エポックごとに，試験データセットの正解値を計算する方法を併用する．図 3 は，第 5 節の MNIST データセットについての実測値のグラフである．損失関数値が減少すると共に，試験データセットの正解値が向上している．

ところで，図 3(a) は正しいと思われる学習プログラム（ProgPC）の振舞いであるが，図 3(b) は欠陥を挿入した学習プログラム（ProgBI）の結果である．両者とも同様な傾向を示していることがわかる．つまり，損失関数値や試験



(a) Probably Correct (ProgPC)

(b) Bug-Injected (ProgBI)

図 3 Loss and Accuracy

データセットに対する正解値の時系列を監視していても，プログラムに欠陥があるか否かの区別がつかない．区別がつかないように，欠陥を混入することができる．

5. 手書き数字認識タスク

5.1 MNIST データセット

データセット多様性の問題を考察する際には学習タスクを決める必要がある．ここでは，手書き数字認識学習に対する方法を，1 桁の手書き数字 1 文字と正解ラベルからなる MNIST データセットを対象として具体的に説明する．

MNIST では，1 枚の手書き数字は 28×28 の大きさを持ち，各ピクセルはグレー画像で 0 から 255 までの値をとる．訓練データセットは 60,000 個，試験データセットは 10,000 個のデータからなる．この学習タスクのニューラル・ネットワークとして，中間層を 50 次元，出力層を 10 次元とした．1 つの入力データは 784 次元となる．

最初に，MNIST データセットの限界データ点を何にすれば良いかを考える．手書き数字認識の面白さは，入力データ $\vec{x}^n$ の属性値から作られるパターンが特定の数字と結論する根拠になることである，そのようなパターンを分離できないと他の数と誤る．既存データと少しだけ異なる新しい手書き数字を追加することを考える．これは，全てのデータを調べて新しい手書き数字を作らなければならないことから現実的でない．また，既存データと全く無関係なデータを追加すると，学習の結果が予測困難になる．表 1 の Inclusive 性を利用することは難しい．

他方，既存データだけを対象とし，特定のピクセル値を変更してグレー階調度を変えると，手書き数字のインクの掠れ具合が変化する状況を作れる．掠れ具合が大きいと，認識に成功するパターンに影響が出る．つまり，表 1 の Additive 性や Multiplicative 性が良いメタモルフィック性になると期待できる．

2 番目に，何をメタモルフィック関係に選ぶかを考える．先の議論から，振舞いオラクルの方法で時系列を調べる．図 1 に関連して説明した仮定が当てはまる統計量を選ぶ必要がある．重みの個数は，39,700（= 784x50 + 50x10）であり，膨大なデータ量となる．一方で，重みは，$\mathbf{V}$ と $\mathbf{W}$ に分かれており，その値の時系列も異なった変化を示すだろう．そこで，$\mathbf{V}$ と $\mathbf{W}$ に区分けし，各々について，エポックごとの平均値と分散を指標の候補とする．

5.2 フォローアップ入力生成

本稿で提案するデータセット多様性に関わる変換関数は，次のような形式をとる．

$$D^{(m+1)} = T(D^{(m)}, G_o^K(\mathbf{V}^{(m)}, \mathbf{W}^{(m)})).$$

今，$\mathbf{V}$ を 10×50 行列とする時，関数 $top(K, o, \mathbf{V})$ は，指定ラベル o（$o = 0, \ldots, 9$）を出力信号とする 50 個の行列要素 v_{oj}（$j = 1, \ldots, 50$）の中から，その重み値がトップ K の要素の添字 j からなる集合 H を返す．次に，H の要素をラベルとして $\mathbf{W}$ に同様の処理を施す．ここで，$h \in H$ をひとつ選ぶとすると，行列要素 w_{hi}（$i = 1, \ldots, 784$）の中から，その値がトップ K の要素の添字 i からなる集合を返す．つまり，$\bigcup_{h \in H} top(K, h, \mathbf{W})$ の集合を得る．このようにして得た添字集合は，指定したラベル o を持つ出力信号に影響を与えやすい入力信号の集まり U を表す．

```
G_o^K(V,W) ≜ {
    H = top(K, o, V);
    return ⋃_{h∈H} top(K, h, W)
}

top(K,ℓ,M) ≜ {
    X = { abs(M_ℓj) }; Idx = ∅;
    repeat K times {
        A_ℓJ = max( X );
        X = X\{ A_ℓJ }; Idx = Idx ∪ { J }
    }
    return Idx
}
```

図 4 Pixel Search

強化対象ピクセルの添字の集合 Blk は，図 4 の関数を用いて求めることができる．関数 T は，上記の処理で得た入力信号の集まり Blk の要素 u について，データセット $D^{(m)}$ が含む $\vec{x}^n$ の u 属性値 x_u^n （$u \in Blk$）を更新する．Multiplicative 性の場合，乗算ファクター F に対して計算した値 $x_u^n \times F$ からなる $\vec{x}^n$ を $D^{(m+1)}$ の要素とする．

なお，上記では，ひとつのラベルを指定し，これを G_o^K と表したが，複数のラベルを指定してもよい．2 つを選ぶ（$G_{o1}{}^{K1} \cup G_{o2}^{K2}$）等に拡張できる．

5.3 メタモルフィック関係の定義

以下，重み行列 $\mathbf{W}$ を対象として考えるが，$\mathbf{V}$ についても同様に定義する．エポック・インデックス e で得られた重み行列の要素を $w_{ji}(e)$ とする．$\mathbf{W}$ は $M \times D$ 行列なので，重みの個数は $M \times D$．$w_{ji}(e)$ の平均は

$$\mu_w(e) = \frac{1}{M \times D} \sum_{j,i} w_{ji}(e)$$

分散は

$$\sigma_w^2(e) = \frac{1}{M \times D} \sum_{j,i} ((w_{ji}(e))^2 - (\mu_w(e))^2)$$

エポックの進行に沿って，平均に対する指標グラフ $g[\mu_w](e)$，分散に対する指標グラフ $g[\sigma_w^2](e)$ を作成する，

MNIST データセットではピクセルは 0 から 255 までの値をとるので，前処理を行って 0 から 1 の値をとるように入力正規化を施すことが多い．いずれにしてもすべての属性 i に対して，$x_i^n \geq 0$．また，教師信号は 0 か 1 なので，$t_k \geq 0$ となる．

ⓒ 2017 Information Processing Society of Japan

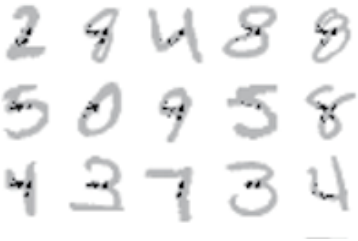
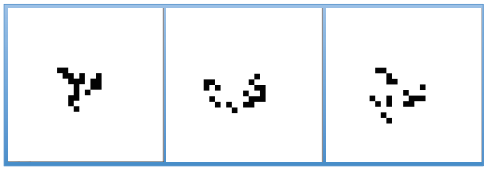


図 5 Multiplicative



(a) $D^{(1)}$ (b) $D^{(2)}$ (c) $D^{(3)}$

図 6 Changes in Dots

重み行列 $\mathbf{W}$ ならびに $\mathbf{V}$ は，正規分布 $Norm(0, std^2)$ にしたがう確率変数値に初期化される．重みが更新されると，$\mathbf{W}$ の各々の値は大きくなり正の収束値に近づくので，平均値 μ_w も大きくなって収束する．一方，重み行列 $\mathbf{V}$ は，10 通りの手書き文字が均等に存在すると仮定すると，各々であると認識する要素の値が大きくなることから，値がばらつくので大きな分散に収束する．したがって，指標グラフ $g[\mu_w](e)$ と $g[\sigma_v^2](e)$ はピークを迎えた後，減少し 0 の近傍で推移する．他方，$g[\mu_v](e)$ と $g[\sigma_w^2](e)$ は，あまり値変化をせずに推移する．実際，MNIST データセットに対しては，この傾向を実験によって確認できる．

以上の定性的な考察から，ピークを持つ $g[\mu_w](e)$ と $g[\sigma_v^2](e)$ を指標グラフとして選ぶ．$\alpha \in \{\mu_w, \sigma_v^2\}$ として，

$$R^T == \bigwedge_\alpha \left(\mathcal{F}(g[\alpha^{(m)}],\ g[\alpha^{(m+1)}]) \leq \epsilon \right)$$

となる．メタモルフィック関係をプログラム実行列に対して定義したことになり，これを振舞いオラクルと呼んだのだった．

5.4 実験

5.4.1 データセット多様性の実験

フォローアップ入力生成の実験結果を報告する．第 1 に，図 5 は，指定ラベルを 4，選択する重みの数を 5 と選んだ場合（G_4^5）を示し，手書き数字 4 の正解値に影響を与えると思われる入力属性のピクセルを表す．さまざまな手書き数字上に黒ドットとして見える．メタモルフィック性として Multiplicative 性を採用し，乗算係数を 4 にした場合の実験結果を示す．乗数係数を大きくすると黒ドットのピクセルを強調した画像になり，逆に，強調されなかったピクセスの認識への貢献度が小さくなる．つまり，強調された少数のピクセスが支配的になり，その他のピクセル（属性値）の効果が薄れるので，全体的に「擦れた」イメージになる．このような手書き画像のデータセットを対象とする場合，一般に解への収束性が悪くなると予想できる．

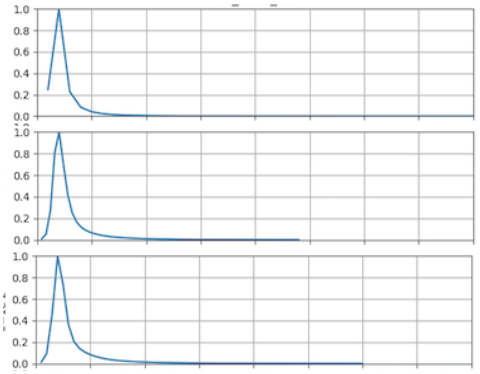


図 7 Indicator Graphs for ProgPC

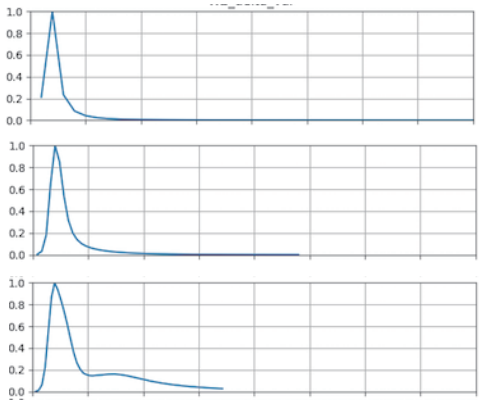


図 8 Indicator Graphs for ProgBI

第 2 に，フォローアップ入力と学習処理を交互に続けて $D^{(m+k)}$ を得る場合の黒ドットの変化を調べた．図 6 は $D^{(0)}$ を MNIST データセットとして，$D^{(1)}$，$D^{(2)}$，$D^{(3)}$ を求めたものである．繰り返しと共に，黒ドット数が増加・拡散し，元のピクセル値の相対的な効果が薄れるので，4 と誤認識される頻度が大きくなる．つまり，学習が難しくなり，収束解ならびに収束に到る傾向が変わる．図 1 に示した理想的な形が崩れると期待される．

5.4.2 振舞いオラクルの実験

次に，データセット多様性と振舞いオラクルに基づくメタモルフィック・テスティングを繰り返す．第 5.3 節で，指標グラフとして $g[\mu_w](e)$ と $g[\sigma_v^2](e)$ を選んだ．以下の議論では，$g[\sigma_v^2](e)$ を具体例として説明する．実験では $g[\mu_w](e)$ も同様な傾向を示しており，先ほどの定性的な予想を裏付けていた．

検査対象プログラムは，図 3 の実行結果を得た 2 つのプログラム，ProgPC と ProgBI，である．各々の $g[\sigma_v^2](e)$ を図 7 ならびに図 8 に示した．

実験の具体的な方法は，図 6 で示した入力フォローアップ生成に準じる．検査対象プログラムをひとつ選び，これを用いて，G_4^3 とし，乗算係数を 4 とした．図 7 および図 8 の上から順に，MNIST データセット $D^{(0)}$，$D^{(1)}$ および $D^{(1)}$ に対する正規化済み指標グラフを示す．以上の 2 つの

実験系列は独立に実施した．

図 7 ならびに図 8 の 1 番目は，それぞれ図 3(a) と (b) の損失関数値と正解率を得た場合に相当し，いずれの指標グラフも理想的な外形に近いことが確認できる．

一方，2 番目，3 番目と入力データセットが「歪む」につれて，指標グラフの形が変わる．図 7 と図 8 から，ProgPC よりも ProgBI の方が形の変化が大きいことが読み取れる．計算した距離の値は，α を σ_v^2 として，ProgPC の場合，

$$\mathcal{F}(g[\alpha^{(0)}],\ g[\alpha^{(1)}]) = 0.000798,$$
$$\mathcal{F}(g[\alpha^{(1)}],\ g[\alpha^{(2)}]) = 0.001544$$

だった．一方，ProgBI の場合，

$$\mathcal{F}(g[\alpha^{(0)}],\ g[\alpha^{(1)}]) = 0.002236,$$
$$\mathcal{F}(g[\alpha^{(1)}],\ g[\alpha^{(2)}]) = 0.010771$$

となった．あきらかに，ProgBI の違いのほうが大きい．歪み方が大きくなっていることが確認できる．

次に，メタモルフィック・テスティングの実施という観点から，これらの実験結果を再考する．テスティングを行う時，ProgPC が存在しないことに注意して欲しい．検査対象は ProgPC なのか ProgBI なのかわからない．しかし，仮に不具合の判定基準を与える ϵ を 0.0020 とすれば，ProgBI についてメタモルフィック関係が壊れており，不具合があると結論することができる．

以上から，判定基準の微小値 ϵ を如何にして決めるかが問題となることがわかる．今回の実験では，同じ検査対象プログラムに対して，学習率を変化させた場合の距離から推測したものである．MNIST データセット $D^{(0)}$ を対象とする場合，学習率を変化させても正規化後の距離の違いは 0.0015 程度だった．そこで，上記では，ϵ を 0.0020 と選んだ．

6. 議論

本稿では，データセット多様性と振舞いオラクルを利用したメタモルフィック・テスティングの方法を提案した．

第 1 に，通常のメタモルフィック・テスティングは，データ多様性に基づいて，入力を工夫する．基本的な発想は，数値計算プログラムの疑似オラクルを得ることだった．本稿のデータセット多様性は，機械学習プログラムは入力が単独データではなく，多数のデータの集まりからなるデータセットであることに着目した．データセット内のデータ分布を系統的に変更する方法を提案した．

一方，文献 [14] の議論からわかるように，表 1 は分類学習で有用な一般的なメタモルフィック性として提案されたものである．実際，機械学習は数値的な最適化プログラムであって，データ多様性の方法が対象とするプログラムである．データセット多様性は，文献 [8][9] で論じたように，効率の良いテスティングを行う限界データ点を導入するということを強調した考え方であって，データ多様性の一種ともいえる．第 5.2 節で論じた方法は，表 1 の multiplicity 性の具体例である．従来のメタモルフィック・テスティングとの違いは，フォローアップ入力生成関数 T の形に現れる．

第 2 に，通常のソフトウェア・テスティングでは，検査対象プログラムが正常終了した時点での計算結果が意図通りか否かを調べる．一方，ニューラル・ネットワークは非凸最適化問題を数値的に解くプログラムとして実現されている．繰り返しアルゴリズムの中で，期待する学習値への収束判定を行う．処理過程で生成する計算値を実行監視することに他ならない．そこで，本稿では，エポック進行での値変化を監視する振舞いオラクルが検査する方法を採用した．

学習対象のパラメータである重みの個数が膨大なことから，個々の重み値ではなく，重み値の統計値を検査対象として，指標グラフを作成した．この振舞いオラクルは，文献 [7] 等の統計的なオラクルとは異なることに注意して欲しい．統計的なオラクルは，同一テスト環境下で実行しても同じ結果とならないプログラムであって，計算結果が確率変数となるようなプログラム，つまり確率的な振舞いを示すプログラムを検査対象とし，計算結果を統計的な方法で解釈する．特に，統計的な仮説検定で，プログラムの不具合があることを論証する方法を論じたものである．本稿の振舞いオラクルで用いる重み値の統計量（平均値と分散）は，一種の要約値であって，統計的な方法によって解釈したわけではない．

本稿では，ニューラル・ネットワークの 2 つのプログラム（ProgPC と ProgBI）を対象とした実験を行った．提案方法によって，ProgPC と ProgBI の振舞いに違いが生じることから，このようなプログラムのソフトウェア・テスティングに利用できる可能性があると論じた．一般に，ソフトウェア・テスティングは，検査対象プログラムの不具合を発見する方法であって，不具合がないことを示す技術ではない [5]．そもそも欠陥がない可能性すらある．だからこそ，欠陥発見の可能性が大きい限界データ点を効率よく求める系統的な方法が重要になる [9]．

本稿で提案したフォローアップ入力生成は，対象とする学習タスクや学習方式の特徴を利用した方法である．実験では，手書き数字認識の問題を対象とした，認識結果に影響を与えるピクセル，つまり入力属性を求める方法は，ニューラル・ネットワークの構造的な性質に着目するものである．ニューラル・ネットワークを用いる他の学習タスクにも応用可能と期待できる．今後，具体的な実験を通しての確認が必要である．

7. おわりに

最近話題になっている「深層学習」は，ニューラル・ネッ

トワークや制限ボルツマン・マシンといった基本的な方式を組み合わせた学習アーキテクチャの総称である [6]．ソフトウェア・テスティングの観点からは，個々の学習方式と全体を統合する学習方式の検査に分けて考えることになろう．一連の検査を進めるテスト過程全体の戦略が重要になると考えられる．

謝辞 実験に協力して下さった今井克則氏（(株) グラッツ）に感謝する．

参考文献

[1] P. Ammann and J.C. Knight : Data Diversity: An Approach to Software Fault Tolerance, *IEEE TC*, vol.37, no.4, pp.418-425, 1988.
[2] P. Ammann and J. Offutt : *Introduction to Software Testing*, Cambridge University Press 2008.
[3] C.M. Bishop : *Pattern Recognition and Machine Learning*, Springer-Verlag 2006.
[4] T.Y. Chen, S.C. Chung, and S.M. Yiu : Metamorphic Testing - A New Approach for Generating Next Test Cases, HKUST-CS98-01, The Hong Kong University of Science and Technology, 1998.
[5] E.W. Dijkstra : The Humber Programmer - ACM Turing Award Lecture, 1972.
[6] I. Goodfellow, Y. Bengio, and A. Courville : *Deep Learning*, The MIT Press 2016.
[7] R. Guderlei, J. Mayer, C. Schneckenburger, and F. Fleischer : Testing Randomized Software by Means of Statistical Hypothesis Tests, In *Proc. SOQUA 2007*, pp.46-54, 2007.
[8] 中島震，Biu Ngoc Hai : テスト不可能プログラム中の準テスト可能コアのテスティング，電子情報通信学会ソフトウェア・サイエンス研究会，札幌，2016.
[9] S. Nakajima and H.N. Bui : Dataset Coverage for Testing Machine Learning Computer Programs, In *Proc. 23rd APSEC*, pp.297-304, 2016.
[10] 中島震 : テスティングからみたニューラル・ネットワーク，SES 2017 — WS2, 早稲田，2017.
[11] S. Nakajima : Generalized Oracle for Testing Machine Learning Computer Programs, In *Proc. 1st FAACS*, Trento, 2017.
[12] 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，日本ソフトウェア科学会第 34 回大会，日吉，2017.
[13] E.J. Weyuker : On Testing Non-testable Programs, *Computer Journal*, 25 (4), pp.465-470, 1982.
[14] X. Xie, J.W.K. Ho, C. Murphy, G. Kaiser, B. Xu, and T.Y. Chen : Testing and Validating Machine Learning Classifiers by Metamorphic Testing, *J. Syst. Softw.*, 84(4), pp.544-558, 2011.

ⓒ 2017 Information Processing Society of Japan