ニューラルネットワークを適用したソフトウェア信頼性評価手法

土肥　正，尾崎　俊治

1. はじめに

ソフトウェアの開発過程における最終段階であるテスト工程において，製品の信頼性を定量的に評価することは開発管理上極めて重要な問題である．そこで，テスト工程において採集されたソフトウェアフォールトの発見時間データから，ソフトウェアに潜在する総フォールト数ならびに当該製品の信頼性を推定することが必要とされている．このような背景から，過去20数年間に渡り，ソフトウェア信頼度成長モデル (Software Reliability Growth Model) と呼ばれる数多くの確率モデルが提案されてきた [1, 2].

しかしながら，(i) ひとつのソフトウェア製品に対するフォールト発見時間（もしくは，ソフトウェア故障発生時間）の時系列データは通常1本しか採集されない，(ii) テスト工程において観測されるフォールト発見時間データ数は必ずしも十分ではない等の理由から，ソフトウェア故障則（もしくはフォールト発見メカニズム）を支配する確率過程を合理的に推定することは困難であり，いくつかの克服すべき問題点が残されていることに注意しなければならない．このような問題点を回避するためのひとつの手段として，ソフトウェアのフォールト発見過程をある種の統計的時系列過程とみなすことにより，テスト工程の将来において観測されるであろうソフトウェアフォールトの発見時間を推定することが考えられる．Singpurwalla and Soyer [3, 4], Chen and Singpurwalla [5], Chatterjee, Misra and Alam [6] はソフトウェアの信頼性予測に自己回帰モデルを適用し，その有効性について検証している．しかしながら，文献 [3-5] で用いられたベイズ推定の枠組みでは事後分布に任意のパラメータが含まれるため，ソフトウェア信頼性評価における統計的推定問題を完全に克服するまでには至っていない．

最近，ソフトウェアのフォールト発見過程を予測するための有力な手段として，人工ニューラルネットワーク (Artificial Neural Network) を応用することが提案されている．Karunanithi, Malaiya and Whitely [7-9], Karunanithi and Malaiya [10, 11], Khoshgoftaar and Szabo [12], Shinohara, Imanishi, Dohi and Osaki [13] は，生体の情報処理メカニズムをモデル化した階層型ニューラルネットワークや相互結合型（リカレント型）ニューラルネットワークをソフトウェアの信頼性予測に適用している．高田, 松本, 鳥居 [14] はソフトウェアフォールト発見時間データだけでなく，ソフトウェア製品の種類や開発工程の規模を特徴付ける他のデータも有機的に結合した予測モデルを構築している．また，篠原, 土肥, 尾崎 [15, 16] は通常の階層型ニューラルネットワークの代わりに GMDH (Group Method of Data Handling) ネットワークを用いることにより，ソフトウェア信頼性評価における自己組織化学習を可能にしている．

本稿の目的は，ソフトウェア信頼性技術における話題の中でも特に，ニューラルネットワークを適用したソフトウェア信頼性評価手法について紹介することである．まず次節において，基本的な階層型ニューラルネットワークの動作規則ならびに学習アルゴリズムについて解説し，3節でニューラルネットワークを用いたソフトウェアフォールト発見時間データの解析例について述べる．4節では，問題点と今後の展望について言及する．

2. ニューラルネットワークによる時系列予測

いま，時刻 0 でテストを開始し，時刻 t において $n\,(\geq 1)$ 個のフォールト発見時間間隔 $\{0 \leq x_1, \cdots, x_n \mid$

どひ　ただし　広島大学工学部第二類
おさき　しゅんじ　広島大学工学部第二類
〒 739-8527 東広島市鏡山 1-4-1

$n\}$ が観測されているものとする．このとき，テスト工程の将来におけるフォールト発見時間間隔 $\tilde{x}_i$ $(i = n+1, \cdots)$ を推定したい．将来におけるフォールト発見時間間隔は過去に観測された a $(2 \leq a \leq n)$ 個の時間間隔データに依存するものと仮定し，図1に示すような，入力層ユニット a (≥ 1) 個，隠れ層 (中間層) ユニット b (≥ 1) 個，出力層ユニット1個の3層ニューラルネットワークによって $\tilde{x}_i$ を予測することを考える．ここでは特に，one look-ahead prediction を行う（すなわち，$\tilde{x}_{n+1}$ を予測する）ことを仮定するが，時刻 t 以降に現われるであろう複数個のフォールト発見時間間隔 $\tilde{x}_{n+1}, \tilde{x}_{n+2} \cdots$ を予測したい場合には，複数個の出力層ユニットをもつ多値出力ニューラルネットワークを用いればよい．

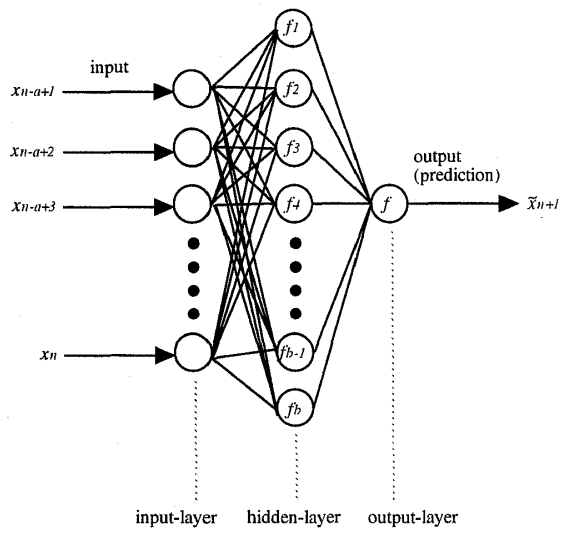

図 1: 階層型ニューラルネットワークの概念図．

図1のニューラルネットワークにおいて，入力層ユニットと隠れ層ユニットの間には結合荷重 w_{ij}^h $(i = 1, \cdots, a, j = 1, \cdots, b)$ が与えられており，隠れ層ユニットへの入力値と出力値は，それぞれ

$$z_j^I = \sum_{i=1}^{a} w_{ij}^h y_{ij}, \tag{1}$$

$$z_j^O = f_j(z_j^I) \tag{2}$$

となる．ここで，$(y_{1j}, \cdots, y_{aj}) = (x_{n-a+1}, \cdots, x_n)$, $j = 1, \cdots, b$ である．関数 $f_j(\cdot)$ はニューロンの特性関数と呼ばれ，微分可能な非減少関数とする．通常，以下のようなシグモイド関数

$$f_j(z) = \frac{1}{1 + \exp\{-(z + \theta_j)\}} \tag{3}$$

が利用されることが多い．ここで，パラメータ θ_j (> 0) はしきい値であり，生理学的根拠から与えられる定数である．最終的に，出力層ユニットからの出力は隠れ層ユニットと出力層ユニット間の結合荷重 w_j^O とシグモイド関数 $f_O(\cdot)$ を用いて，

$$\tilde{x}_{n+1} = f_O\Big(\sum_{j=1}^{b} w_j^O z_j^O\Big) \tag{4}$$

となり，n (≥ 1) 個めから $n+1$ (≥ 1) 個めまでのフォールト発見時間間隔に対する推定値としてみなすことができる．

ネットワークの学習を行うための最も簡便かつ実用的な方法として誤差逆伝搬法が知られている（例えば [17]）．これは，入力データに対するネットワークの出力結果と教師信号（すなわち，観測値 $x_1, \cdots, x_n$）との2乗誤差を減らす方向に結合荷重としきい値の値を変更する学習アルゴリズムである．換言すれば，各入力パターンに関する2乗誤差の勾配ベクトルを計算し，その逆方向に学習パラメータを更新するという，いわゆる最急降下法を利用するものである．

3. ソフトウェアフォールト発見時間データの解析

実際のソフトウェアテスト工程において観測された136個のフォールト発見時間データ [18] を使用し，逐次的に将来におけるフォールト発見時刻を推定するものとする．ここでは推定モデルとして，入力層ユニット10，中間層ユニット10，出力層ユニット1の階層型ニューラルネットワーク (NN) と，代表的なソフトウェア信頼度成長モデルである (i) Jelinski and Moranda (JM) モデル [19] (ii) Schick and Wolverton (SW) モデル [20], (iii) Moranda (MR) モデル [21] をそれぞれ比較する．特に，階層型ニューラルネットワークでは入出力応答にシグモイド関数を仮定するものとし，シグモイド関数に含まれるしきい値パラメータと各ユニット間の初期結合荷重に対する初期値は先行実験および一様乱数によりそれぞれ定めるものとする．教師信号として観測されたソフトウェアフォールト発見時間データを用い，ネットワークの学習には2節で述べた古典的な誤差逆伝搬法を採用する．

Jelinski and Moranda モデル [19] はソフトウェア信頼度成長モデルの中で最も基本的かつ代表的なモデルとして知られており，ソフトウェアフォールト発見時間間隔を表す非負で独立な確率変数列を X_i $(i =$

$1,2,\cdots,N)$, X_i の実現値を x_i, X_i のハザード（故障）率を $h_i(x_i)=\Pr\{X_i\in[x_i,x_i+dx_i)\mid X_i>x_i\}/dx_i$ とすれば，

$$h_i(x_i)=\phi(N-i+1) \tag{5}$$

によって与えられる．ここで，N (>0) はテスト開始前にソフトウェア内に潜在する初期フォールト数を表すパラメータであり，ϕ (>0) は残存フォールト1個当りの発見率を表す．Jelinski and Moranda モデルにおいて，$i-1$ 番目から i 番目のフォールトが発見されるまでの平均時間である MTBF (Mean Time Between Failures) は $\mathrm{E}[X_i]=1/\phi(N-i+1)$ となるので，ソフトウェアフォールト発見時間間隔データ $\{x_1,x_2,\cdots,x_n\mid n\}$ から最尤法により未知パラメータ N と ϕ を推定し，MTBF の推定値を得る．一方，Schick and Wolverton モデル [20] と Moranda モデル [21] はそれぞれ次のように与えられる．

$$h_i(x_i)=\phi(N-i+1)x_i, \tag{6}$$

$$h_i(x_i)=\phi k^{i-1}. \tag{7}$$

ここで，k $(0<k<1)$ はハザード率が幾何級数的に減衰する様子を描写するパラメータである．

図 2 に，$n=61$ の時点からソフトウェアの累積 MTBF を逐次予測した結果を表す．ここで，図中の JM1 と JM2 は，観測時点における全てのソフトウェアフォールト発見時間間隔データ $(a=n)$ を最尤推定に用いた場合と，観測時点からさかのぼってニューラルネットワークの入力層ユニット数と同数 $(a=10)$ のデータを用いた場合をそれぞれ意味している．この図から，ニューラルネットワークによる予測値は実際の累積フォールト発見時間間隔を若干少なめに見積もっているが，Jelinski and Moranda モデルではかなり大きめの推定値を与えていることがわかる．同様に，Schick and Wolverton モデル，Moranda モデルとの比較結果を図 3 と図 4 に示す．予測の傾向はほぼ Jelinski and Moranda モデルの場合と同じであり，ニューラルネットワークに基づいた方法は真値と比べて楽観的な予測を行うことが結論付けられる．

最終的に，$n=61$ から $n=136$ までの各時点での予測値に対する平均2乗誤差 $\{(\text{実測値}-\text{推定値})/\text{実測値}\}^2$ を算出した結果を表 1 に示す．ここで，ニューラルネットワークの予測値に対する平均2乗誤差が $\mathrm{NN}=4.550\times10^{-2}$ となったのに対し，Jelinski and Moranda モデルでは，$\mathrm{JM1}=3.760\times10^{-1}$, $\mathrm{JM2}=9.201\times10^{-2}$ となった．一方，Schick and R. Wolverton モデル，Moranda モデルでは，$\mathrm{SW1}=1.037$, $\mathrm{SW2}=3.229\times10^{-1}$, $\mathrm{MR1}=1.379\times10^{-1}$, $\mathrm{MR2}=7.879\times10^{-2}$ となり，ニューラルネットワークによる予測は通常のソフトウェア信頼度成長モデルと比べて良好な結果を得ることが示される．

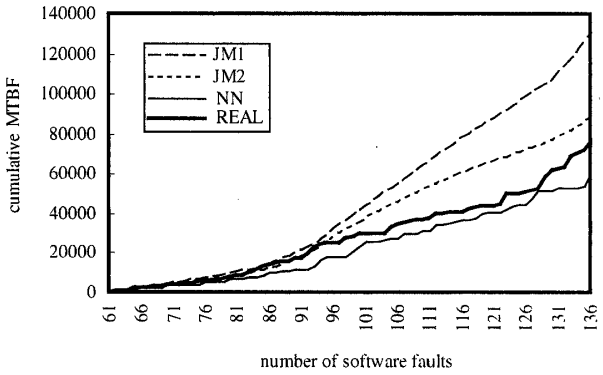

図 2: 累積 MTBF の比較 (NN v.s. JM Model).

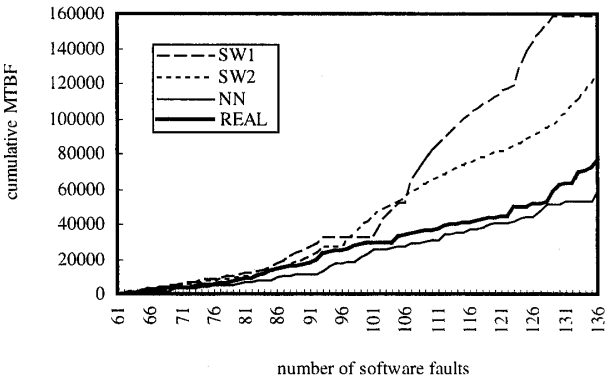

図 3: 累積 MTBF の比較 (NN v.s. SW Model).

4. いくつかの問題点と今後の課題

本稿では，ニューラルネットワークを用いたソフトウェアの信頼性評価法について大まかに解説した．ここでは最も基本的なニューラルネットワークである階層型ニューラルネットワークを適用したが，相互結合型ネットワーク等の異なるアーキテクチャや改良されたネットワーク学習アルゴリズムを用いることにより予測精度が飛躍的に向上する点にも留意すべきであろう．また，本稿では紙面の都合上ふれることができ

なかったが，ソフトウェアをテスト段階から運用段階に移行する時期を決定するソフトウェア最適リリース問題はソフトウェアの開発管理上重要な問題として認識されており，ニューラルネットワークを用いた解法が極めて有効であることが示されている [22-25].

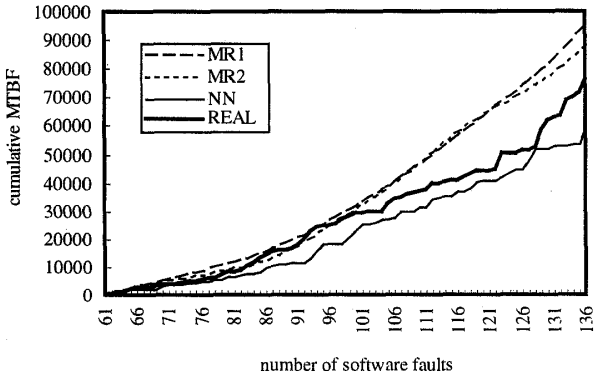

図 4: 累積 MTBF の比較 (NN v.s. MR Model).

表 1: 平均 2 乗誤差.

prediction model	mean squared error
NN	4.550×10^{-2}
JM1	3.760×10^{-1}
JM2	9.201×10^{-2}
SW1	1.037
SW2	3.229×10^{-1}
MR1	1.379×10^{-1}
MR2	7.879×10^{-2}

これまで述べてきたように，ニューラルネットワークに基づいた予測手法は，従来のソフトウェア信頼度成長モデルに比べて概ね良好な結果を与える傾向にある [7-13, 15, 16]. 現在まで，実に 100 を超えるソフトウェア信頼度成長モデルが提案されているが，実際に数多くの予測モデルの中から最良モデルを選択する手続きは必ずしも容易ではなく [1, 2]，何らかの統一的な予測手法が必要とされていた．その意味で，ニューラルネットワークによるソフトウェアフォールト発見時間の予測は，従来から用いられてきた確率モデルと比較してソフトウェアのデバッグ過程に対する物理的意味づけを必要としない点で大変便利である．

ニューラルネットワークのもつ優れた情報処理能力をソフトウェアの信頼性予測に適用することは大変興味深い反面，いくつかの問題点が指摘されていることも事実である．例えば，

(i) ニューラルネットワークの設計ないし学習には往々にして試行錯誤的な要素が含まれているため，実際に信頼性予測ツールとしてシステム化することが困難である場合が多い．すなわち，大域的最適解を保証するような学習アルゴリズムの改良や最適なネットワークサイズの決定法などを詳細に吟味する必要がある

(ii) 実際のテスト工程においてはフォールト発見時間のグループデータしか記録されない場合が多い．そこで，ヘビサイド関数を特性関数にもつ離散型ニューラルネットワークを使用する等の工夫が必要とされる

(iii) ある定められた期間内にソフトウェア故障が発生する確率を表すソフトウェア信頼度を，ニューラルネット計算上において厳密な意味で如何に定義するか

等が問題点として挙げられている．今後，ニューラルネットワークによるソフトウェアの信頼性予測手法のさらなる進展を期待したい．

謝辞

本研究の一部は文部省科学研究費奨励研究 (A) 課題番号 11780331 の助成の下で行われたものである．

参考文献

[1] J. D. Musa, A. Iannino and K. Okumoto: *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[2] 山田茂: ソフトウェア信頼性モデル– 基礎と応用, 日科技連, 1994.

[3] N. D. Singpurwalla and R. Soyer: "Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications", *IEEE Trans. Software Eng.*, Vol. SE-11, pp. 1456–1464, 1985.

[4] N. D. Singpurwalla and R. Soyer: "Nonhomogeneous autoregressive processes for tracking (software) reliability growth, and their Bayesian analysis", *J. Roy. Statist. Soc.*, Vol. B-54, pp. 145–156, 1992.

[5] Y. Chen and N. D. Singpurwalla: "A non-Gaussian Kalman filter model for tracking software reliability", *Statist. Sinica*, Vol. 4, pp. 535–548, 1994.

[6] S. Chatterjee, R. B. Misra and S. S. Alam: "Prediction of software reliability using an auto regressive process ", *Int. J. Sys. Sci.*, Vol. 28, pp. 211–216, 1997.

[7] N. Karunanithi, Y. K. Malaiya and D. Whitley: "Prediction of software reliability using neural networks", *Proc. 2nd Int. Symp. Software Reliab. Eng.*, pp. 124–130, IEEE Computer Society Press, 1991.

[8] N. Karunanithi, D. Whitley and Y. K. Malaiya: "Using neural networks in reliability prediction", *IEEE Software*, Vol. 9, pp. 53–59, 1992.

[9] N. Karunanithi, D. Whitley and Y. K. Malaiya: "Prediction of software reliability using connectionist models", *IEEE Trans. Software Eng.*, Vol. SE–18, pp. 563-574, 1992.

[10] N. Karunanithi and Y. K. Malaiya: "The scaling problem in neural networks for software reliability prediction", *Proc. 3nd Int. Symp. Software Reliab. Eng.*, pp. 76–82, IEEE Computer Society Press, 1992.

[11] N. Karunanithi and Y. K. Malaiya: "Neural networks for software reliability engineering", *Handbook of Software Reliability Engineering*, M. R. Lyu (ed.), pp. 699–728, McGraw-Hill, New York, 1996.

[12] T. M. Khoshgoftaar and R. M. Szabo: "Predicting software quality, during testing, using neural network models: a comparative study", *Int. J. Reliab. Qual. Safe. Eng.*, Vol. 1, pp. 303-319, 1994.

[13] Y. Shinohara, M. Imanishi, T. Dohi and S. Osaki: "Software reliability prediction using neural network technique", *Proc. 2nd Australia-Japan Workshop on Stochastic Models in Engineering, Technology & Management*, R. J. Wilson, D. N. P. Murthy and S. Osaki (eds.), pp. 564–571, Technology Management Centre, The University of Queensland, Brisbane, 1996.

[14] 高田義弘, 松本建一, 鳥居宏次: "ニューラルネットを用いたソフトウェア信頼性予測モデル", 電子情報通信学会論文誌, Vol. J77-DI, pp. 454–461, 1994.

[15] 篠原康秀, 土肥正, 尾崎俊治: "GMDH ネットワークによるソフトウェアのテスト進捗度に対する予測評価", 電子情報通信学会論文誌, Vol. J80-A, pp. 1982–1988, 1997.

[16] 篠原康秀, 土肥正, 尾崎俊治: "予測平方和を用いた改良型GMDH ネットワークによるソフトウェア故障データの解析", ソフトウェアシンポジウム '97 論文集, ソフトウェア技術者協会, pp. 205–211, 1997.

[17] 馬場則夫, 小島史男, 小澤誠一: ニューラルネットの基礎と応用, 共立出版, 1994.

[18] J. D. Musa: "Software reliability data", *Technical Report in Rome Air Development Center*, New York, 1979.

[19] Z. Jelinski and P. B. Moranda: "Software reliability research", *Statistical Computer Performance Evaluation*, W. Freiberger (ed.), pp. 465–484, Academic Press, New York, 1972.

[20] G. J. Schick and R. Wolverton: "An analysis of competing software reliability models", *IEEE Trans. Software Eng.*, Vol. SE-4, pp. 104–120, 1978.

[21] P. B. Moranda: "Prediction of software reliability during debugging", *Proc. 1975 Annu. Reliab. Maintainance Symp.*, pp. 327–332, 1975.

[22] 土肥正, 西尾泰彦, 篠原康秀, 尾崎俊治: "ニューラルネットワークを用いたソフトウェア最適リリース問題の幾何学的解法", 電子情報通信学会論文誌, Vol. J81-A, pp. 110–118, 1998.

[23] Y. Shinohara, Y. Nishio, T. Dohi and S. Osaki: "An optimal software release problem under cost rate criterion: artificial neural network approach", *J. Qual. Maintenance Eng.*, Vol. 4, pp. 236–247, 1998.

[24] 篠原康秀, 土肥正, 尾崎俊治: "GMDH ネットワークに基づいたソフトウェア出荷スケジュールの生成アルゴリズム", 日本信頼性学会誌, Vol. 21, pp. 2–10, 1999.

[25] T. Dohi, Y. Nishio and S. Osaki: "Optimal software release scheduling based on artificial neural networks", to appear in *Ann. Software Eng.*.