뉴로-퍼지 소프트웨어 신뢰성 예측
(Neuro-Fuzzy Approach for Software Reliability Prediction)

이 상 운 †
(Sang-Un Lee)

요 약 본 논문은 주어진 고장 데이타로부터 소프트웨어의 신뢰성 예측력 향상을 위해 뉴로-퍼지 시스템 연구를 수행하였다. 다른 소프트웨어로부터 수집된 10개의 고장 수 데이타와 4개의 고장시간 데이타에 대해 규칙의 수를 변경시키면서 다음 단계 예측을 실험하였다. 뉴로-퍼지 시스템의 예측력을 보이기 위해 다음 단계 예측에 대해 비교하였다. 실험 결과 뉴로-퍼지 시스템은 다양한 소프트웨어에 잘 적용되었다. 또한 널리 사용되고 있는 신경망과 통계적 소프트웨어 신뢰성 성장 모델의 예측력과 견줄 정도의 좋은 결과를 얻었다.

Abstract This paper explores neuro-fuzzy system in order to improve the software reliability predictability from failure data. We perform numerical simulations for actual 10 failure count and 4 failure time data sets from different software projects with the various number of rules. Comparative results for next-step prediction problem is presented to show the prediction ability of the neuro-fuzzy system. Experimental results show that neuro-fuzzy system is adapt well across different software projects. Also, performance of neuro-fuzzy system is favorably with the other well-known neural networks and statistical SRGMs.

1. 서 론

시험과 운영 단계에서 획득된 고장 데이타로부터 소프트웨어 신뢰성을 추정하는데 일반적으로 소프트웨어 신뢰성 모델이 사용되고 있다. 소프트웨어 신뢰성 모델의 예측력 (Predictability 또는 Generalization Ability)은 과거와 현재의 고장 발생 현상으로부터 미래의 고장 발생 현상을 예측하는 모델의 능력을 의미한다. 신뢰성 평가에 사용되는 고장 데이타는 시간 계 (Time Domain) 데이타로 고장 수 (Failure Count)와 고장시간 (Failure Time) 형태로 수집된다. 고장 수 데이타는 i 번째 단위시간인 t_i 에서 발견된 고장 수를, 고장시간 데이타는 i 번째 고장이 발견된 시간을 기록하는 경우이다.

대부분의 소프트웨어 신뢰성 성장 모델 (SRGMs : Software Reliability Growth Models)은 소프트웨어에 내재된 결함 특성과 고장의 통계적 발생 과정에 관한 가정에 기반을 두고 소프트웨어에 대한 전반적인 신뢰성 평가 및 예측을 제공하며, 또한 최적의 소프트웨어 양도 시점을 결정하는데 사용된다. 이 부류의 모델들은 특정 소프트웨어의 개발 및 운영 환경에 대한 다양한 가정을 반영한 다수의 모수 (Parameters)들을 포함하고 있다. 최적의 소프트웨어 신뢰성 모델 선택에 관해 Abdel-Ghaly, Chan과 Littlewood [1], Brocklehurst, Chan, Littlewood와 Snell [3] 이 연구하였으나 모든 소프트웨어 시스템에 적용 가능한 하나의 보편적인 모델이 없는 실정이다. 특정 SRGM 선택은 소프트웨어 신뢰성 평가를 향상시키는데 매우 중요하다. 따라서, 소프트웨어 개발 및 운영 환경에 대한 어떤 가정을 요구하지 않는 소프트웨어 신뢰성 모델에 대한 연구가 필요하다.

만약, 소프트웨어 시스템의 과거 고장 이력에 기반을 두고 자체의 모델을 개발하는 시스템을 갖고 있다면 그와 같은 가정은 고려하지 않아도 된다. 신경망 (Neural Networks)의 경우 사전 정보인 소프트웨어의 환경과 조건에 대한 가정을 고려하지 않고 단지 관찰된 고장 이력만을 입력으로 요구한다. Karunanithi, Whitley와

† 비 회 원 : 경상대학교 전자계산학과
sangun_lee@hanmail.net
논문접수 : 1998년 7월 4일
심사완료 : 1999년 7월 13일

Malaiya [5],[6]는 feedforward networks (FFNs)과 Jordan network에 대한 예측력을 평가하고, 고장 수 데이타에 대해 잘 알려진 통계적 SRGMs 와 모델의 예측 성능을 비교하였으며, Park, Lee와 Park [14]은 고장시간 데이타에 대해 유사한 연구를 수행하였다. 신경망을 이용한 소프트웨어 신뢰성 예측의 경우, 예측력 향상을 위해서는 최적의 은닉 뉴런 (Hidden Neurons)의 개수 결정, 학습을 언제 종료할 것인가 (Stopping Rule), 연결 강도 (Connection Weights)에 대한 랜덤한 초기 값 설정으로 인해 학습시마다 신경망의 결과가 다르게 나타남으로 인해 동일한 데이타에 대해 몇 번의 학습을 수행 (Trials)하는 것이 최적인가, 훈련제도 (Training Regime)는 어떤 유형으로 사용할 것인가, 데이타 표현 (Data Representation)은 어떻게 할 것인가, 데이타를 훈련 (Training Data Set)과 검증 (Validation Data Set), 시험 데이타 집합(Testing Data Set)으로 어떤 비율로 분할해야 모델의 예측력이 최적일 것인가 등의 문제점이 발생되며, 현재까지 명확하게 제시된 규칙은 없고 여러 번의 수행 결과를 비교해 그중 최적의 결과를 나타내는 모델을 선정하는 Trial-and-Error 방법이 널리 사용되고 있다.

신경망은 주어진 데이타로부터 망의 구조에 대한 사전 정보를 전혀 가지지 못한 원점에서 학습을 하는데 반해 뉴로-퍼지 시스템 (Neuro-Fuzzy System)은 사전 지식 (Prior Knowledge)을 가진 상태에서 학습을 하는 장점을 갖고 있다. 뉴로-퍼지 시스템은 신경망 이론에서 유도된 학습 알고리즘으로 훈련되는 퍼지 시스템으로 적절한 뉴로-퍼지 시스템을 구축하기 위해서는 주어진 문제에 대해 맴버쉽 함수 (모수)와 규칙 (구조)이 정의되어야만 한다. 뉴로-퍼지 시스템은 신경망의 단점을 극복하기 위해 최근에 연구되고 있는 분야로 뉴로-퍼지 시스템을 활용한 소프트웨어 신뢰성 예측에 관한 연구는 현재까지 진행되지 않고 있다. 따라서, 본 논문에서는 고장 데이타에 대한 소프트웨어 신뢰성 예측을 위해 뉴로-퍼지 시스템을 적용할 수 있음을 제시하고자 한다. 2장에서는 관련 연구에 대해 개략적으로 살펴보고, 3장에서는 뉴로-퍼지 시스템 구현 방법에 대해, 4장에서는 소프트웨어 신뢰성 예측을 위해 뉴로-퍼지 시스템을 적용하는 방법과 모델의 예측 성능을 비교 분석하여 본다.

2. 관련 연구 및 연구 배경

신경망은 모수 추정과 미래의 출력을 예측하기 위해 적용되고 있다. 예측을 위해 잘 알려진 망으로는 FFNs과 recurrent networks이나 소프트웨어 신뢰성 예측에는 단지 몇 건의 연구만 수행되었다. Karunanithi, Whitley와 Malaiya [5],[6]는 신경망을 이용한 소프트웨어 신뢰성 성장에 대해 모델링하였으며, FFNs과 partial recurrent networks (PRNs)인 Elman Network과 Jordan network이 소프트웨어 신뢰성 예측에 적용 가능함을 보였다.

(t_i, m_i), $i=1, 2, \cdots, n$ 데이타가 소프트웨어 시험중 수집되었다고 가정하자. 여기서, m_i는 시간 t_i 까지 발견된 누적 고장 수이다. (t_i, m_i)를 고장 수 데이타라 칭한다. Karunanithi, Whitley와 Malaiya [5],[6]는 훈련을 위해 주어진 (t_i, m_i), $i=1, 2, \cdots, l$ 로 m_{l+p}, $p=1, 2, \cdots$를 예측하기 위해 FFNs 과 PRNs을 연구하였으며, 2가지 훈련제도를 고려하였다. (1) 일반 훈련제도 (각 입력 t_i는 목표 m_i와 관련), (2) 예측 훈련제도 (각 입력 t_{i-1} 는 목표 m_i와 관련).

FFNs과 PRNs의 예측력이 평가되고 잘 알려진 통계적 SRGMs와 비교되었다. 예측력을 평가하고 비교하기 위해 14개의 다른 소프트웨어 시스템으로부터 수집된 데이타가 사용되었다. 14개의 고장 수와 고장시간 데이타에 대해 표 1에 간략하게 기술하고 있다. 그들은 FFNs과 PRNs이 다양한 데이타에 잘 적용되었으며, 특히 마지막 단계 (End-Point)에 대한 예측력이 보다 좋음을 나타냈다. 그러나 다음 단계 (Next-Step) 예측에 대해서는 특별히 좋은 결과를 얻지 못했다. 14개 데이타 중 Data2, Data11, Data12와 Data13은 (i, s_i), $i=1, 2, \cdots, n$로 구성되어 있으며, 여기서 i 는 고장 발생 일련번호이고 s_i는 i 번째 고장시간이다. 이러한 데이타를 고장시간 데이타라 칭한다. Karunanithi, Whitley와 Malaiya [5],[6]는 고장시간 데이타에 대해 신경망의 입력으로 고장시간을, 목표로는 고장 발생 일련번호를 적용하였다.

고장시간 데이타의 경우, 통계적 고장 발생 과정의 관찰자료로 고장시간을 이용해 고장 발생 일련번호를 분석하고 예측하는 것은 타당하지 않다. 우리가 예측하고자 하는 것은 고장 발생 일련번호가 아니라 고장이 발생한 시간에 대한 통계적 과정이기 때문이다. 따라서 Park, Lee와 Park [14]은 고장시간 데이타에 대해 다음과 같은 4개의 훈련제도를 고려하였다. (1) 각 입력 i 는 목표 x_i와 관련, (2) 각 입력 i 는 목표 x_{i+1}와 관련, (3) 각 입력 i 는 목표 s_i와 관련, (4) 각 입력 i 는 목표 s_{i+1}와 관련. 여기서 $i=1, 2, \cdots$에 대해 $x_i = s_i - s_{i-1}$이다. 훈련제도 (1)과 (3)은 일반 훈련제도이며, (2)와 (4)는 예측 훈련제도이다. FFNs에 대한 훈련

제도 (3)과 (4)가 소프트웨어 신뢰성 예측에 적합함을 보였으며 예측력은 훈련 데이타 크기에 커다란 영향을 받지 않음을 밝혔다.

표 1 14개의 고장 데이타

데이타명	참고문헌	LOC	고장수	데이타크기	적용분야
Data1	[7]	1,000	27	14	Class Compiler Project
Data2	[8][9]	21,700	136	136	Realtime Control
Data3	[13]	40,000	46	21	On-line Data Entry
Data4	[13]	1,317,000	328	17	Database Application
Data5	[13]	35,000	279	10	Hardware Control
Data6	[16]	240,000	3,207	13	Application Software
Data7	[18]	870,000	535	109	Realtime Control
Data8	[17]	200,000	481	111	Monitoring and Realtime Control
Data9	[17]	14,5000	55	199	Railway Interlocking
Data10	[17]	90,000	198	16	Monitoring and Realtime Control
Data11	[2]	10,000	118	118	Flight Dynamic
Data12	[2]	22,500	180	180	Flight Dynamic
Data13	[2]	38,500	213	213	Flight Dynamic
Data14	[18]	not known	266	46	Realtime Control

좋은 예측 모델이란 미래의 행위를 잘 예측하고, 단순하며, 다양한 조건에 적용해야 한다. 이러한 요구조건을 충족시키는 것 중의 하나가 적응필터이며, Park, Lee와 Park [15]은 적응 필터를 예측 필터로 변형시켜 소프트웨어 신뢰성 예측을 연구하였다. t_i 시간까지 발견된 누적 고장 수는 $\hat{m}_i = \sum_{k=1}^{d} w_k m_{i-k} + b$로, i 번째 고장이 발견된 시간 $\hat{s}_i = \sum_{k=1}^{d} w_k s_{i-k} + b$로 예측되며, 이는 각각 m_{i-k}와 s_{i-k}, $k=1,2,\cdots,d$의 선형결합 형태의 입력을 의미한다. 예측필터는 데이타 크기가 클 경우 (Data 7 ~ 9) 예측력이 가장 좋았으며, 데이타 크기가 적을 경우 모든 소프트웨어에 가장 적합한 모델이 없음을 보였다. 따라서, 예측 필터는 단순하면서도 좋은 예측 결과를 나타내 좋은 모델의 요구조건을 충족함을 밝혔다.

이러한 다양한 연구에도 불구하고, 신경망을 적용하는데 있어서 발생되는 가장 중요한 단점중의 하나는 주어진 데이타로부터 모델의 예측력이 뛰어난 최적의 은닉 뉴런 수를 결정할 수 있는 사전 정보를 갖지 못한다는 것이다. 따라서, 신경망 보다는 모델을 구축하기 위해 필요한 사전 정보를 사용할 수 있는 뉴로-퍼지 시스템이 소프트웨어 신뢰성을 예측하는데 적용 가능성을 제시한다. 3장에서는 뉴로-퍼지 시스템을 구현하는 방법을 간략히 살펴본다.

3. 뉴로-퍼지 시스템

주어진 문제에 대해 적절한 퍼지시스템을 확인하기 위해서는 맴버쉽 함수와 규칙수가 정의되어야만 하며, 이는 사전 지식에 의해, 학습에 의해, 또는 이 2가지를 합성한 방법에 의해 수행된다. 만약, 퍼지 시스템에 있는 지역 정보를 사용하고 지역적인 변경을 수행하기 위해 학습 알고리즘이 적용된다면 이 시스템을 뉴로-퍼지 시스템이라 칭한다. 즉, 뉴로-퍼지 시스템은 주어진 샘플 데이타를 처리하여 시스템의 모수인 맴버쉽 함수의 형태 모수와 퍼지 규칙을 결정하기 위해 신경망 이론에서 유도된 학습 알고리즘을 사용하는 퍼지 시스템으로 입-출력 데이타 쌍 $(t_i, m_i), i=1,2,\cdots,l$이 주어졌을 때 은닉층이 1개인 FFN의 경우를 그림 1에서 표현하고 있으며, 다음과 같은 속성을 가지고 있다. (1) 신경망과는 연결 강도, 작동 함수 (Activation Function)가 다르며 학습 절차는 국부 정보 (Local Information)에만 작동하여 주어진 퍼지 시스템에서 단지 국부적 변경만을 수행한다. (2) 특수한 3-계층 FFN으로 볼 수 있으며, 뉴런은 신경망에서 사용되는 작동 함수 대신 퍼지 $t-$norms (합집합), $t-$conorms (교집합), multiply 또는 min 사용하며, 첫 번째 층은 입력 변수를, 2번째 층 (은닉층)은 퍼지 규칙을 표현하며, 3번째 층은 출력 변수를 표현한다. 또한 퍼지 집합은 퍼지 연결 강도로 변환된다. (3) 일부분만 주어진 훈련 데이타로 미지의 $n-$차원 함수를 근사시킬 수 있다.

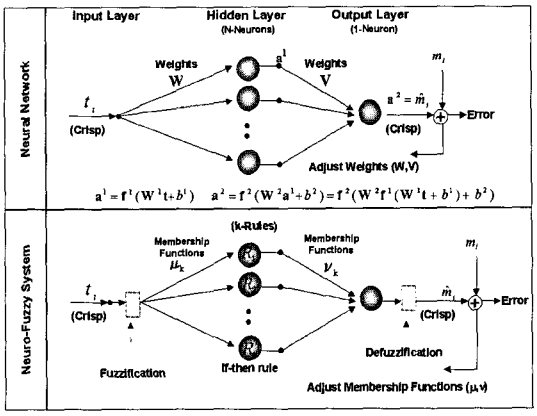


그림 1 신경망과 뉴로-퍼지시스템의 3-계층 표현

뉴로-퍼지 시스템은 주어진 데이타만을 가지고 시스템을 구성하는 기법이며, 차이점은 신경망은 원점에서부터, 뉴로-퍼지 시스템은 사전 지식을 가진 상태에서 시

스템을 구성하는 것이다. 신경망의 FFN의 경우 이전 층의 모든 뉴런에서 다음 층의 한 뉴런으로 모두 연결이 되어 있으며 이 연결 강도는 모두 다른 강도를 가지며 학습과정 중에 모두 다르게 변경된다. 이에 반해 뉴로-퍼지 시스템은 몇몇 연결들이 항상 동일한 퍼지 강도 (맴버쉽 함수)를 가지고 연결되어 주어진 입력이 퍼지화 되며, 학습과정 중에도 항상 동일한 값으로 변화된다. 또한 은닉 뉴런 대신 규칙이 사용되고, 규칙 (추론)을 거친 값이 역퍼지화된 후 출력되어 주어진 함수에 근사한 값을 출력한다. [10],[11],[12]

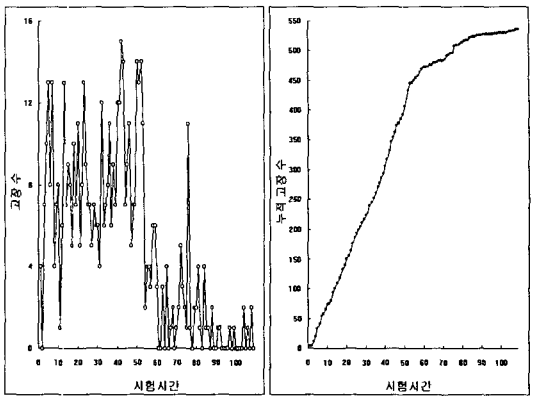
표 2 모델 개발 단계

단계		퍼지 관점	신경망 관점
1	추출	주요 입력 변수	주요 입력 변수
2	결정	규칙수, 맴버쉽 함수	은닉 뉴런수, 연결 강도
3	조절	맴버쉽 함수	연결 강도
4	제거	미사용 규칙	미사용 뉴런

퍼지추론의 출력 맴버쉽 함수 형태에 따라 Mamdani와 Sugeno 형태로 구분된다. Mamdani 형태는 출력 맴버쉽 함수가 퍼지 집합으로 표현되며 각 출력 변수에 대한 퍼지 집합이 역퍼지화 과정을 거쳐야 한다. 이에 반해 Sugeno 형태는 출력 맴버쉽 함수가 선형 또는 상수항으로 표현된다. 뉴로-퍼지 시스템 중 함수 근사에 적용되고 있는 소프트웨어로는 ANFIS (Adapted Network-based Fuzzy Inference System)과 NEFPROX (NEuro-Fuzzy function aPROXimation)가 있다. ANFIS는 Sugeno 형태의 퍼지시스템이며, 이에 반해 NEFPROX는 Mamdani 형태의 퍼지시스템이다. [4],[11],[12],[13]

4. 뉴로-퍼지 적용 소프트웨어 신뢰성 예측

소프트웨어 시험 중 수집된 고장 수 데이타 (단위 시간당 고장 수와 누적 고장 수)의 일반적인 유형이 그림 2에 표현되어 있다. 단위 시간당 고장 수는 일반적으로 누적 고장 수에 비해 복잡한 비선형 함수 형태로 명확한 패턴이 나타나지 않기 때문에 적합한 모델을 결정하기 어려우며 모델의 예측력도 나쁘게 나타난다. 고장시간 데이타의 경우도 유사한 형태를 취하며, 고장간 시간 데이타의 유형이 명백한 패턴 형태를 나타내지 않는다. 따라서 고장 수 데이타에 대해서는 누적 고장 수에 대해, 고장시간 데이타에 대해서는 고장간 시간이 아닌 고장시간 데이타에 대한 예측을 연구 대상으로 한다.



(a) 단위 시간당 고장 수 (b) 누적 고장 수

그림 2 소프트웨어 고장 유형 (Data7)

뉴로-퍼지 시스템 설계시, 주어진 소프트웨어에 최적인 규칙 수에 대한 사전 정보를 알고 있지 못하므로 규칙 수를 2개에서 시작하여 30개까지 1개씩 추가시키면서 신경망에서 유도된 Gradient Descent와 LMS (Least Mean Square error) 알고리즘을 사용하여 시스템을 학습시켜 시스템의 출력과 목표 값 (Target)간의 오차가 최소가 되도록 한다.

시스템의 훈련제도로는 Karunanithi, Whitley와 Malaiya [5],[6] 가 제안한 일반 훈련제도와 예측 훈련제도를 사용한다. 신경망과 마찬가지로 뉴로-퍼지 시스템도 주어진 입-출력 데이타만으로 모델을 구축하는 능력을 갖고 있기 때문에 최소한의 훈련 데이타 크기를 3개로 정하고 다음 단계에 대한 예측을 수행한다. 따라서 예측을 수행하는 시점은 4부터 주어진 소프트웨어 고장 데이타의 최대 크기까지가 된다. 2개 훈련제도 각각에 대해 최적으로 예측력을 나타내는 규칙 수를 택해 뉴로-퍼지 소프트웨어 신뢰성 예측 모델을 구축한다.

다음 단계 예측에는 2장에서 기술된 10개의 고장 수 데이타와 4개의 고장시간 데이타가 사용되었다. 시스템의 훈련을 종료하는 방법으로 Early Stopping 기법을 사용하였다. 시스템의 훈련을 수행하기 위해 주어진 데이타를 훈련 데이타와 시험 데이타 집합으로 나누고, 훈련 데이타를 다시 훈련 데이타 집합과 검증 데이타 집합으로 나눈다. Early Stopping (또는 Stopped Training, Optimal Stopping)기법이란 훈련 데이타 집합을 사용하여 시스템을 훈련시키고, 검증 데이타를 사용해 훈련을 종료하는 최적의 시점을 결정하는 방법으

로, 전형적으로 훈련데이타의 오차는 훈련을 진행하면서 계속적으로 감소하는데 비해 검증 데이타 집합의 오차는 어느 시점까지는 감소하다가 갑자기 증가하는 경향을 나타내고 있다. 따라서, 검증 데이타의 오차가 갑자기 증가 (Overfitting)하기 시작하기 바로 이전에 시험을 종료하는 것이 시스템 훈련에 사용되지 않은 제 3의 데이타 집합 (시험 데이타)의 예측력을 최적으로 판단하는 시점으로 알려져 있다. 훈련 데이타 집합과 검증 데이타 집합의 최적의 분할 비율에 대해서는 명확한 규칙이 없으며, 일반적으로 Single-Sampling Statistics, Split-Sample or Hold-Out Method, Bootstrapping, Cross-Validation (Leave-One-Out, v-Fold) 등 다양한 방법이 있다. Cross-validation은 적은 데이타에 대해 split-sample 방법 보다 향상된 성능을 발휘하나 resampling에 근거하여 시스템의 오차를 추정하는 방법으로 시스템을 여러번 다시 훈련시켜야 하는 단점이 있다. 따라서, 본 연구에서는 split-sample 형태를 취하나 훈련과 검증 데이타의 분할 비율은 $l-1:1$의 비율로 leave-one-out validation 형태를 취하는 방법에 한정하여 모델의 예측력을 분석하고자 한다. 즉, $1,2,\cdots,l-1$을 훈련 데이타 집합으로, l 번째 데이타를 검증 데이타 집합으로 설정하고 시스템을 ANFIS 소프트웨어를 통해 시스템을 훈련시켜 $l+1$번째 데이타 (시험 데이타 집합)의 예측력에 대한 실험을 수행하였다. 그러나 선택된 데이타 분할 비율, 샘플링 방법, 샘플링 횟수에 따라 시스템의 예측력에는 차이가 발생할 수 있으며, 다양한 기법이 적용될 수 있다.

4.1 고장 수 데이타

주어진 고장 수 데이타 $(t_i, m_i), i=1,2,\cdots,n$에 대해, l을 훈련 데이타 크기로, r은 예측 단계로 가정하자. 예를 들면, 다음 단계 예측은 $r=1$, 마지막 단계 예측에 대해서는 $r=(n-l)$이 된다. 신경망 또는 뉴러-퍼지 시스템의 특징은 주어진 데이타로부터 스스로 학습하는 능력을 갖고 있다는 점이다. 따라서 뉴러-퍼지 시스템이 학습 능력을 갖게 하기 위해서는 필요한 최소한의 훈련 데이타가 필요하다. 뉴러-퍼지 시스템을 훈련시키는 최소한의 데이타 크기는 Karunanithi, Whitley와 Malaiya [6]의 신경망을 이용한 소프트웨어 신뢰성 예측에 적용한 바와 같이 3 (즉, $l=3$)으로 설정한다. 주어진 훈련 제도에 대해 다음 단계 예측을 수행한 후 Karunanithi, Whitley와 Malaiya [6]의 신경망과 통계적 소프트웨어 신뢰성 성장모델의 예측 결과와 비교하고자 한다. 다음 단계 예측 절차는 다음과 같이 수행된다.

단계 (1) 규칙 수 가정.

단계 (2) 훈련 데이타 크기 l 과 훈련제도 결정.

단계 (3) 학습 알고리즘과, 훈련 데이타 (t_i, m_i), $i=1,2,\cdots,l$을 사용하여 훈련 수행.

단계 (4) t_{l+j}, $j=1,2,\cdots,r$ 시간을 입력하여 j번째 미래의 누적 고장 수, $\hat{m}_{l+j}$, 예측과 상대오차 $e_{lj}=\frac{(\hat{m}_{l+j}-m_{l+j})}{m_{l+j}}\cdot 100$ 계산.

단계 (5) l을 1 증가, $l=n-r$이 될 때까지 (2) ~ (3) 단계 수행.

단계 (6) l 에 대해 e_{lj} 의 평균 계산.

l 에 대한 e_{lj}의 평균은 $\bar{e}_{\cdot j}$로 표기되며, 이는 평균 상대 예측 오차 (the average relative prediction error : AE) 이다.

Karunanithi, Whitley와 Malaiya [5],[6]는 예측력 측도로서 l 에 대해 e_{l1} 과 $e_{l(n-l)}$을 고려하였다. 이것은 각각 다음 단계와 마지막 단계에 대한 예측 오차이다. 미래의 몇 단계 앞을 예측할 것인가는 주관적 관점에 따라 차이가 발생되나 마지막 단계 예측은 데이타의 개수에 따라 차이가 발생되며, 실제 적용에 있어서도 시험을 종료하는 시점을 명확히 판단할 수 없다. 우리의 주요 관심은 소프트웨어가 주어진 시간 동안 (다음 단계) 고장 없이 작동하는가 여부이므로 본 연구에서는 다음 단계 예측에 한정하여 모델의 예측력을 분석하여 본다. 그러나 다양한 r 이 적용 가능하다.

훈련제도 각각에 대해 규칙 수를 증가시키면서 단계 (2) ~ (6)이 수행되고 다음 단계 예측 값인 출력 $\bar{e}_{\cdot 1}$'s 이 표현되었다. 시스템의 다음 단계 평균 예측 오차가 최소가 되는 규칙 수가 선택되었다. Data7에 대해 최적인 규칙을 사용한 경우의 실제 발견된 누적 고장 수 m_i와 뉴러-퍼지 시스템의 예측된 누적 고장 수 $\hat{m}_i$를 그림 3에, 뉴러-퍼지 시스템의 예측 시점별 상대오차인 AE는 그림 4에 나타내었다.

그림 3에서 뉴러-퍼지 시스템의 누적 고장 수에 대한 예측 결과 (일반과 예측 훈련제도로 예측된 누적 고장 수)가 시험 시간이 예측 초반 단계인 4 ~ 15와 60 ~ 90 사이에서 다소 저하되었을 뿐 전반적으로 실제의 누적 고장 수 m_i에 근접되어 예측력이 좋음을 알 수 있다. 그림 4에서 예측 초반 (데이타의 크기가 충분하지 않은 경우)에는 예측 오차가 많이 발생되나 점차 안정화되어 시험시간이 90 이후에는 예측 오차가 거의 발생되지 않는, 즉, 뉴러-퍼지 시스템이 다음 단계의 실제 발견된 누적 고장 수를 거의 정확히 예측 가능함을 알 수 있다.

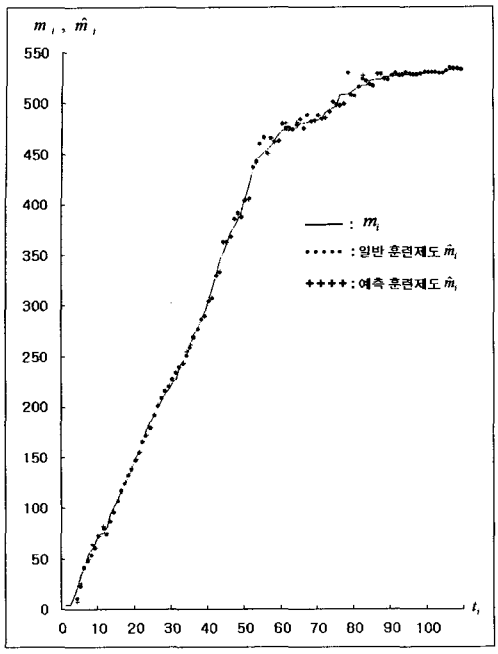


그림 3 Data7에 대한 m_i 와 $\hat{m}_i$ (규칙 수 = 20)

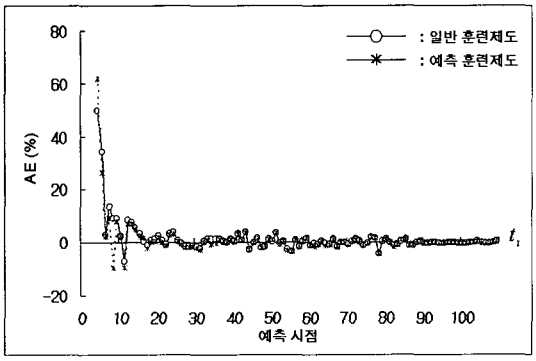


그림 4 Data7에 대한 AE (규칙 수 = 20)

표 3은 10개의 고장 데이타에 대해 해당 데이타의 모든 예측 시점에 대한 상대오차를 평균하여 최소의 상대오차를 나타내는 규칙의 수 (최적의 규칙 수)를 선정한 것이다. 표에서 알 수 있듯이 뉴러-퍼지 시스템에 사용되는 규칙 수는 주어진 문제 (데이타)의 복잡성에 의존하며, 시스템의 예측력은 선택된 규칙 수의 크기에 영향을 받으며, 최적의 규칙 수를 수리적으로 얻기 위한 일반적인 해답이 없음을 알 수 있다.

표 3 10개 고장 수 데이타의 최적의 규칙 수

Data Set	Data 1	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 14
일반훈련제도	15	19	8	27	23	20	7	5	4	30
예측 훈련제도	9	5	11	11	28	20	7	6	2	29

일반 훈련제도와 예측 훈련제도에 대한 결과 $\bar{e}_{\cdot 1}$'s 가 Karunanithi, Whitley와 Malaiya [6], Park, Lee와 Park [14]의 결과와 비교한 자료는 표 4에 표현되어 있다. ( )는 각 소프트웨어에 대한 해당 모델의 예측 정확성 순위이다. 표 4에서 특정 모델은 특정 소프트웨어에만 적합하며, 일반적인 가장 좋은 모델을 선정하기가 어려움을 알 수 있다.

표 4 고장 수 데이타에 대한 $\bar{e}_{\cdot 1}$

모델		다음 단계 평균 상대오차 (Rank)									
		Data 1	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 14
뉴러-퍼지 시스템	일반 훈련 제도	7.17 (5)	10.35 (10)	**4.59 (1)**	7.27 (6)	**2.58 (1)**	2.35 (3)	3.01 (3)	4.63 (2)	8.01 (11)	6.81 (11)
	예측 훈련 제도	8.46 (9)	10.56 (11)	7.37 (9)	6.05 (2)	4.13 (7)	2.33 (2)	2.82 (2)	4.92 (3)	6.77 (9)	6.86 (12)
신경망 모델	예측 필터	7.53 (7)	10.56 (12)	6.27 (8)	8.48 (7)	4.47 (9)	**1.61 (1)**	**2.47 (1)**	**3.54 (1)**	9.59 (12)	3.53 (5)
	FFN -일반 훈련 제도	9.37 (11)	8.44 (9)	5.28 (3)	10.0((10)	4.33 (8)	3.66 (11)	6.56 (10)	11.82 (11)	5.90 (7)	4.56 (8)
	FFN -예측 훈련제도	5.61 (2)	6.84 (5)	4.64 (2)	6.95 (5)	4.51 (10)	3.74 (12)	5.24 (9)	6.72 (5)	7.45 (10)	4.80 (9)
	JordanNet -일반 훈련제도	6.79 (4)	6.84 (5)	8.84 (11)	**5.09 (1)**	5.25 (12)	2.97 (8)	9.11 (12)	9.72 (7)	4.08 (4)	4.86 (10)
	JordanNet -예측 훈련제도	**4.66 (1)**	6.03 (2)	6.11 (6)	8.67 (8)	5.24 (11)	2.90 (6)	3.73 (5)	6.41 (4)	**3.22 (1)**	4.50 (7)
통계적 소프트웨어 신뢰성 성장모델	Logarithmic	7.33 (6)	7.78 (7)	5.93 (4)	6.42 (4)	3.47 (2)	2.88 (5)	4.47 (8)	10.20 (8)	3.75 (3)	3.24 (3)
	Inverse Polynomial	9.21 (10)	6.17 (3)	7.95 (10)	9.71 (9)	3.59 (5)	2.92 (7)	4.45 (7)	9.09 (6)	4.99 (5)	**3.13 (1)**
	Exponential	6.67 (3)	7.85 (8)	6.01 (5)	6.15 (3)	3.51 (3)	2.50 (4)	4.26 (6)	13.06 (12)	3.28 (2)	3.52 (4)
	Power	11.1[illegible] (12)	6.55 (4)	9.40 (12)	23.3((12)	3.51 (3)	3.19 (10)	7.06 (11)	11.36 (10)	5.53 (6)	3.20 (2)
	Delayed S-shape	8.03 (8)	**5.99 (1)**	6.25 (7)	10.9((11)	3.63 (6)	3.07 (9)	3.39 (4)	10.86 (9)	6.57 (8)	3.55 (6)

4.2 고장시간 데이타

주어진 고장시간 데이타 $(i, s_i), i=1,2,\cdots,n$에 대해, Park, Lee와 Park [15]은 신경망을 활용해 고장시간 데이타에 대한 소프트웨어 신뢰성 예측 모델을 제시하

였으며, 훈련에 사용된 최소한의 데이타 크기를 약 20%로 설정 한 후 데이타 크기를 5씩 증가시켜 가면서 다음 단계의 고장 발생시간을 얼마나 정확히 예측하는 가를 살펴보았다. Park, Lee와 Park [15]의 결과와 비교하기 위해 본 논문에서도 동일한 방법으로 설정하고, 실험을 수행한다. 다음 단계 예측 절차는 다음과 같이 수행된다.

단계 (1) 규칙 수 가정.
단계 (2) 훈련 데이타 크기 l 과 훈련제도 결정.
단계 (3) 학습 알고리즘과, 훈련 데이타 (i, s_i), $i=1,2,\cdots,l$을 사용하여 훈련 수행.
단계 (4) $l+j$, $j=1,2,\cdots,r$ 시간을 입력하여 j번째 미래의 누적 고장시간, $\hat{s}_{l+j}$,예측과 상대오차 $e_{lj}=\frac{(\hat{s}_{l+j}-s_{l+j})}{s_{l+j}}\cdot 100$ 계산.
단계 (5) l을 5 증가, $l=n-r$이 될 때까지 (2) ~ (3) 단계 수행.
단계 (6) l 에 대해 e_{lj} 의 평균 계산.

Data2에 대해 최적인 규칙을 사용한 경우의 s_i와 $\hat{s}_i$를 그림 5에, AE를 그림 6에서 보여주고 있으며. 표 5는 4개의 고장시간 데이타에 대해 선택된 최적의 규칙 수를 나타내고 있다. 일반 훈련제도와 예측 훈련제도에 대한 결과 $\bar{e}_{\cdot 1}$'s 가 Park, Lee와 Park [15]의 결과와 비교한 자료는 표 6에 표현되어 있다. 그림 5에서 뉴러-퍼지 시스템의 고장시간에 대한 예측 결과가 시험 시간이 경과할 수록 다소 저하되었을 뿐 전반적으로 실제의 고장 발생시간 s_i에 근접되어 예측력이 좋음을 알 수 있다. 그림 5의 시스템 예측 결과에 대한 상대오차를 그림 6에서 보여주고 있으며, 시험 시간이 경과되면서 발생된 고장시간 데이타의 개수가 많아져 실제 상대 오차는 적어짐을 보이지만 시험 종료 시점에 가까워지면서 오차는 커짐을 알 수 있다. 표 5와 6에서도 고장 수 데이타와 동일하게 최적의 규칙 수는 주어진 문제의 복잡성에 기인하며, 특정 모델이 특정 데이타에 적합함을 알 수 있다.

4.3 예측력 결과 분석

뉴러-퍼지 시스템이 소프트웨어 신뢰성 예측에 적용 가능성을 판단하기 위해 널리 사용되고 있는 신경망과 통계적 소프트웨어 신뢰성 성장 모델의 신뢰성 예측 정확성을 비교하여 보자. 여기서는 고장 수와 고장 시간 데이타에 대해 Karunanithi, Whitley와 Malaiya [6]가 제시한 모델의 예측력 측도를 사용하고자 한다. 이 방법

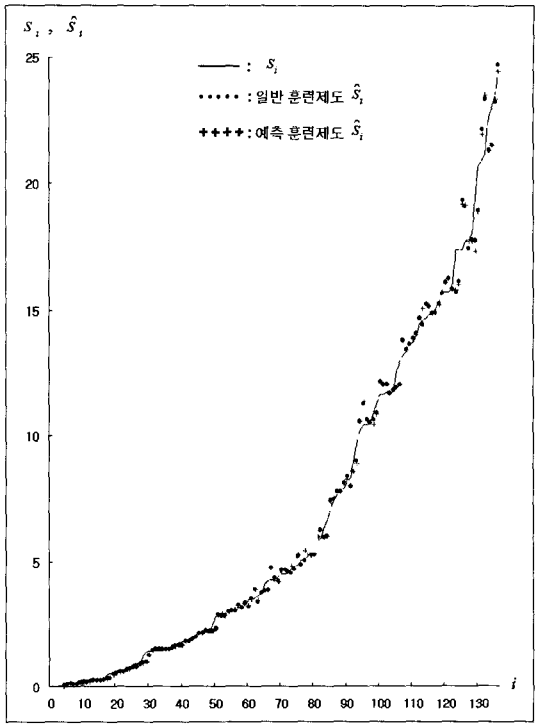


그림 5 Data2에 대한 s_i 와 $\hat{s}_i$ ($r=1$)

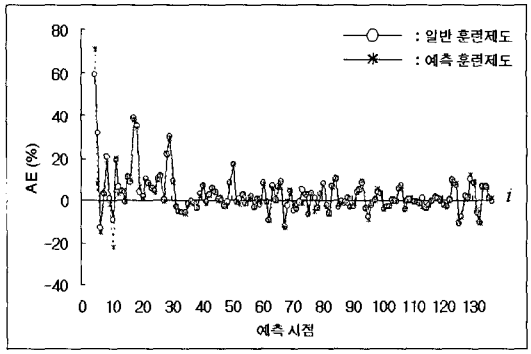


그림 6 Data2에 대한 AE ($r=1$)

표 5 고장시간 데이타의 최적의 규칙 수 ($r=5$)

Data Set	Data2	Data11	Data12	Data13
일반 훈련제도	23	10	17	30
예측 훈련제도	22	11	22	29

표 6 고장시간 데이타에 대한 $\bar{e}_{\cdot,1}$ ($r=5$)

모델		다음 단계 예측 평균 상대오차 (Rank)			
		Data2	Data11	Data12	Data13
뉴러-퍼지 시스템	일반 훈련제도	3.26 (4)	2.32 (2)	2.45 (5)	**1.50 (1)**
	예측 훈련제도	3.26 (4)	2.35 (3)	2.43 (4)	1.54 (3)
신경망 모델	예측 필터	**1.80 (1)**	**1.57 (1)**	**1.25 (1)**	1.94 (5)
	FFN -일반 훈련제도	2.28 (2)	3.32 (4)	2.28 (2)	1.58 (4)
	FFN -예측 훈련제도	2.58 (3)	3.32 (4)	2.38 (3)	1.51 (2)

은 각각의 데이타에 대해 각 모델의 정규화된 상대오차 $NAE^m = \frac{AE^m}{AE^m_{max}}$ 를 구한 후 각 모델에 대해 $R_m = \sum_{i=1}^{\#\ of\ data\ set} NAE^m$ 을 구하여 모델의 예측력 순위를 결정한다. 여기서 m은 신뢰성 예측에 사용된 소프트웨어 수이다. 모델의 예측력 결과가 표 7에 제시되어 있다.

표 7 모델의 예측력 순위

모델		고장 수 데이타		고장시간 데이타	
		R_m	순위	R_m	순위
뉴러-퍼지 시스템	일반 훈련제도	6.0563	5	3.4720	3
	예측 훈련제도	6.6041	9	3.4935	4
신경망 모델	예측 필터	6.0440	4	2.5352	1
	FFN -일반 훈련제도	7.3378	11	3.4444	2
	FFN -예측 훈련제도	6.3673	8	3.5412	5
	JordanNet -일반 훈련제도	7.0873	10		
	JordanNet -예측 훈련제도	5.6756	1		
통계적 소프트웨어 신뢰성 성장모델	Logarithmic	5.8658	2		
	Inverse Polynomial	6.2974	6		
	Exponential	5.9040	3		
	Power	7.8297	12		
	Delayed S-shape	6.3028	7		

실험 결과 다음과 같은 결론을 얻을 수 있었다.

(1) 고장 수와 고장시간 데이타 모두에 대해 뉴로-퍼지 시스템의 예측력이 널리 사용되고 있는 특정 신경망과 통계적 소프트웨어 신뢰성 성장 모델보다 좋은 결과를 나타내었다. (2) 시스템의 예측 오차와 최적의 규칙 수는 훈련제도와 고장 데이타의 복잡한 비선형적 특성에 크게 의존한다. (3) 뉴러-퍼지 시스템이 주어진 비선형 함수에 대한 함수 근사 능력을 갖고 있음을 Nauck과 Kruse [12]가 제시하고 있으며, 그림 3과 5에서 소프트웨어 시험 결과 수집된 누적 고장 수 또는 고장발생 시간 자료도 비선형 함수 형태를 취함을 알 수 있다. 본 실험 결과 주어진 데이타의 비선형 함수에 대한 뉴러-퍼지 시스템의 함수 근사 학습 능력 뿐 아니라 미지의 데이타에 대한 예측 (일반화) 능력도 좋음을 알 수 있다. 따라서, 뉴러-퍼지 시스템이 소프트웨어 신뢰성 예측에 적용 가능함을 제시할 수 있다.

5. 결 론

본 논문은 관찰된 고장 데이타로부터 소프트웨어 신뢰성을 예측하기 위해 뉴로-퍼지 시스템을 연구하였다. 본 연구에는 뉴로-퍼지 시스템을 적용함에 있어서 사전 정보가 없다는 가정하에 Sugeno 형태인 선형 또는 상수로 출력 맴버쉽 함수를 사용하였으며, 10개의 고장 수 데이타와 4개의 고장시간 데이타 실험 결과 소프트웨어 신뢰성 예측에 뉴로-퍼지 시스템의 적용 타당성을 보였다. 뉴로-퍼지 시스템의 성능은 고장 데이타에 대해 잘 알려진 특정 신경망과 통계적 소프트웨어 신뢰성 성장 모델보다 좋은 예측 결과를 나타내었다. 따라서, 사전 정보를 활용할 수 없는 신경망의 단점을 보완한 뉴로-퍼지 시스템 적용은 소프트웨어 신뢰성 예측에 있어서 다음 단계 예측력이 좋고, 다양한 환경과 조건의 소프트웨어에 적용 가능함을 알 수 있었다.

신경망에서의 훈련 데이타 분할 비율과 은닉 뉴런수 문제와 마찬가지로 뉴로-퍼지 시스템도 모델의 예측 정확성은 훈련 데이타의 적절한 분할 비율과 규칙 수에 크게 영향을 받는다. 본 실험에서는 훈련과 검증 데이타 집합의 분할 비율을 $1-1:1$로 설정하였으나 다른 비율을 적용할 경우 모델의 예측력에 영향을 미칠 수 있다. 좋은 예측력을 얻기 위해 최적의 데이타 분할, 샘플링 방법, 샘플링 횟수에 대한 연구와 더불어 데이타 크기와 규칙 수 사이의 관계에 대해 추후 연구가 될 것이다.

참 고 문 헌

[1] A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," IEEE Trans. Software Eng., Vol. SE-12, pp. 950-967, 1986.

[2] B. M. Anna-Mary, "A Study of the Musa Reliability Model," M.S. dissertation, Univ. Maryland, 1980.
[3] S. Brocklehurst, B. Y. Chan, B. Littlewood and J. Snell, "Recalibrating Software Reliability Models," IEEE Trans. Software Eng., Vol. 16, pp. 458-470, 1990.
[4] J. -S. R. Jang, "ANFIS : Adaptive-Network- Based Fuzzy Inference System," IEEE Trans. on Systems, Man., and Cybernetics, Vol. 23, No. 3, pp. 665-685, 1993.
[5] N. Karunanithi, D. Whitley and Y. K. Malaiya, "Prediction of Software Reliability Using Connectionist Models," IEEE Trans. Software Eng., Vol. 18, pp. 563-574, 1992.
[6] N. Karunanithi, D. Whitley and Y. K. Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software, Vol. 18, pp. 53-59, 1992.
[7] K. Matsumoto, T. Inoue, T. Kikuno, and K. Torii, "Experimental Evaluation of Software Reliability Growth Models," Proc. IEEE Conf. FTCS-18, pp. 148-153, 1988.
[8] J. D. Musa, A. Lannino, and K. Okumoto, *"Software Reliability : Measurement, Prediction, Application,"* McGraw-Hill, New York, 1987.
[9] J. D. Musa, "Software Reliability Data," Technical Report, Data and Analysis Center for Software, Rome Air Development Center, Griffins AFB, New York, 1979.
[10] D. Nauck, "Neuro-Fuzzy Systems : Review and Prospects," Proc. 5th European Congress on Intelligent Techniques and Soft. Computing, Aachen, pp. 1044-1053, 1997.
[11] D. Nauck and R. Kruse, "A Fuzzy Neural Network Learning Fuzzy Control Rules and Membership Functions by Fuzzy Error Backpropagation," Proc. IEEE Int. Conf. Neural Networks, ICNN'93, San Francisco, pp. 1022-1027, 1993.
[12] D. Nauck and R. Kruse, "A Neuro-Fuzzy Approach to Obtain Interpretable Fuzzy Systems for Function Approximation", Proc. IEEE Conf. on Fuzzy Systems, Anchorage, AK, pp. 1106-1111, 1998.
[13] M. Ohba, "Software Reliability Analysis Models," IBM H. Res. Develop., Vol. 28, pp. 428-443, 1984.
[14] J. Y. Park, S. U. Lee, and J. H. Park, "Neural Network Modeling for Software Reliability Prediction from Failure Time Data," Journal of Electrical Eng. and Information Science, Vol. 4, No. 4, pp. 533-538, 1999.
[15] J. Y. Park, S. U. Lee, and J. H. Park, "Predictive Filter for Software Reliability Prediction," Submitted to Journal of Electrical Eng. and Information Science, 15. Sep. 1999.
[16] M. L. Shooman, "Probablistic Models for Software Reliability Prediction," Statistical Computer Performance Evaluation, New York Academic, pp. 485-502, 1972.
[17] Y. Thoma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution," IEEE Trans. on Software Eng., Vol. 15, pp. 345-355, 1989.
[18] Y. Thoma, H. Yamano, M. Ohba, and R. Jacoby, "Parameter Estimation of the Hyper-Geometric Distribution Model for Real Test/Debug Data," Dept. Computer Science, Tokyo Inst. Tech., Tech. REP. 901002, 1990.

이 상 운

1983년 ~ 1987년 한국항공대학교 항공전자공학과 학사. 1995년 ~ 1997년8월 경상대학교 전자계산학과 석사. 1998년 ~ 현재 경상대학교 전자계산학과 박사과정. 관심분야는 소프트웨어 공학(소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성), 신경망, 퍼지