인공신경망을 이용한 소프트웨어 개발공수 예측모델에 관한 연구

전 응 섭†

요 약

소프트웨어 개발공수(Efforts)에 관한 연구는 그동안 상당히 많이 이루어져 있으나, 대부분 기존의 알고리즘 모델과 통계적 접근방법에 의한 모델에 한정되어 있다고 할 수 있다. 또한 이들 연구는 주로 외국의 사례를 대상으로 한 것이어서 국내의 소프트웨어 개발환경에 적용하기에는 예측력과 적응도 등의 여러 문제가 제기되고 있다. 따라서 본 논문에서는 보다 현실적이고 실용적인 소프트웨어 개발공수의 예측모델로서 백프로퍼게이션 알고리즘을 이용한 신경망 예측모델을 제시하고, 이 모델의 예측결과와 기존의 모델인 COCOMO 그리고 회귀분석에 의한 예측결과들을 통계적으로 비교 분석하여 신경회로망의 우수한 예측력을 검증하였다. 이러한 분석의 결과를 토대로 보다 예측력이 좋고 사용자가 쉽게 모델링하여 사용할 수 있는 정교한 신경망 모델을 제시하고자 한다.

Using Artificial Neural Network for Software Development Efforts Estimation

Eung Sup Jun †

ABSTRACT

In the research area of estimation of the software development efforts, a number of researches have been accomplished in order to control the costs and to make software more competitive. However, most of them were restricted to the functional algorithm models or the statistic models. Moreover, since they are dealing with the cases of foreign countries, the results are hard to apply directly to the domestic environment for the efficient project management because of lack of accuracy, fitness, flexibility and portability. Therefore, it is appropriate to suggest and propose a new approach supported by artificial neural network which is composed of back propagation and feed-forward algorithms to improve the exactness of the efforts estimation and to advance practical uses. In this study, the artificial neural network approach is used to model the software cost estimation and the results are compared with the revised COCOMO and the multiregression model in order to validate the superiority of the model.

1. 서 론

소프트웨어(S/W)개발 프로젝트를 관리하는데 가장 중요한 기법중의 하나가 프로젝트 계획수립이다. 소프트웨어 개발 프로젝트의 계획수립은 일반적으로 투입되는 인적자원의 정확한 예측을 요구하는데 이에 대한 예측이 용이하지가 않다.

기존의 소프트웨어 개발 투입공수(Efforts)에 관한 예측모델로 여러가지의 모델이 제시되고 있는데, 그 중 가장 보편적으로 제시되고 있는 예측기법으로 COCOMO 모델과 통계적 예측모델인 다중 회귀모델을 들 수가 있다.

그러나 이러한 모델들도 소프트웨어 개발의 고

† 정 회 원 : 인덕전문대학 사무자동화과 조교수
논문접수 : 1995년8월18일, 심사완료 : 1995년1월3일

유한 환경적 특성의 한계점을 극복하는데 난점이 있어서, 각 조직의 특성과 현실을 반영하지 못하고 있고, 예측력의 신뢰도 및 설명도도 부족하여 일반적으로 수용하기에는 무리가 있는 것이 사실이다.

본 연구에서는 예측력과 적응력이 약한 기존의 모델보다 좀더 신뢰성이 있고 예측력이 우수한 모델로 개발하기 위해 신경망 모델에 의한 S/W 개발 투입공수(Efforts)예측 모델을 구축한다. 또한 실증데이터를 적용하여 예측력을 검정하고 모델의 적합도를 시험한후 예측에 관련된 제반 문제의 해결 접근방법으로 보다 예측력이 좋은 모델을 제안하고자 한다.

이러한 새로운 예측모델을 제안하기 위해 국내외 K공사에서 1982년 부터 1989년동안 기 개발된 115개의 소프트웨어 개발업무의 실제관련 데이터를 수집하여 신경망 모델의 학습과 예측에 사용하였다.

2. 소프트웨어개발 예측모델의 유형 및 특성

2.1 소프트웨어개발 예측 기법

소프트웨어 개발시 계획 단계에서 개발 기간, 개발 인원 및 개발비를 정확하게 예측하는 일은 프로젝트 관리의 가장 중요한 시작이라고 할 수 있는데,이를 위한 여러가지 예측 기법과 모델들이 제시되었다.

1) 전문가 판단법

여러명의 전문가가 모여서 여러번의 검토를 거쳐서 S/W개발 비용을 추정하는 방법으로 DELPHI기법 같은 전문가의 합의체의 도움을 얻는 방법이다. 참가한 전문가의 의견만 반영되고 경우에 따라 편견이 발생될 수 있다.

2) 유추법

이전에 수행되었던 유사한 프로젝트의 경험을 바탕으로 신규 프로젝트의 S/W개발 비용을 추정하는 방법으로 유사한 프로젝트 경험이 없는 신규 프로젝트의 비용추정이 어렵고 과거의 프로젝트 경험치의 대표적 적절성 여부가 고려되어야 한다.

3) 파킨슨 법칙

일은 이용 가능한 양을 충분히 활용하도록 확대된다는 법칙에 따르는 것으로 결국 비용추정치와 이용 가능한 자원이 같게 되도록 원용된다.

4) Price-to-Win

프로젝트를 성공시키는데 필요하다고 인정되는 금액을 S/W개발비로 추정하는 방법으로서 일반적으로 S/W개발비가 초과 발생되는 형태로 추정되므로 권장되지 못한다.

5) 하향식 방법

프로젝트 수행에 필요한 전체 비용을 S/W제품의 전체적인 특성에서 유도하여 여러 하위 부분체계로 나누어 가는 방법으로 효율적인 방법은 되나 안정적이지 못하다.

〈표 1〉 예측기법의 비교
〈Table 1〉 Comparison of estimation techniques

방 법	장 점	단 점
알고리즘모델	• 객관적이고 반복 및 분석이 가능한 공식 • 민감성 분석에 효율적이고 경험을 객관적으로 조정	• 투입자료의 주관성과 예외적인 환경의 평가 • 미래가 아닌 과거 시점
전문가 판단법	• 대표적업무 평가, 상호작용 및 예외적인 환경 평가	• 편견과 불완전한 사고
유추법	• 대표적 경험에 기반	• 경험의 대표성 문제
파킨슨 법칙	• 어떤 경험과 연관되어 있음	• 서투른 실습의 강요
Price-to-Win	• 일반적인 계약체결	• 일반적으로 보다 큰 비용 발생
하향식 방법	• 시스템 수준의 중점과 효율성	• 상세하지 않은 기반과 불안정성
상향식 방법	• 보다 상세한 기반과 안정적이며 개별적인 참여 조성	• 시스템 수준과 비용간의 불합리성, 보다많은 노력 요구

6) 상향식 방법

S/W개발 작업의 개별요소를 세부적으로 추정하여 그 결과를 집계하여 전체 프로젝트 비용을 추정하는 방법으로 상세한 근거 첨부가 가능하며 안정적인 추정방법이다. 그러나 개별요소 추정으로 많은 노력이 필요하나 표준 품셈이 제정되어 있으면 좋은 방법이다.

7) 알고리즘 모델

S/W개발비용 예측에 중요한 비용 요소들을

수학적인 방법으로 공식을 작성하여 추정하는 방법이다. 다른 방법에 비해 많은 강점을 가지고 있으며 프로젝트에 영향을 미치는 변수들에 비교적 객관적인 판단을 할 수 있다.

2.2 소프트웨어 개발공수 예측모델

알고리즘 모델들은 소프트웨어의 개발환경 계수들이 비정형적으로 구해진 후,이를 프로그램 규모의 함수로 주어지는 개발노력값에 가중치로서 곱해진다.이들 모델들의 단점은 개발환경 계수들이 사용자의 경험에 의해 구해진다는 점과 개발될 소프트웨어의 규모가 주어져야만 개발노력 값들을 구할 수 있다는 것이다. 그럼에도 불구하고 아직까지는 가장 보편적이고 비교적 합리적이라는 이유로 여러 모델이 개발제시되고 있다.

1) SDC(Nelson, 1965)

소프트웨어 개발시 미치는 특성들로 104개의 비용요인을 적용하여 회귀분석을 이용해서 특히 중요한 14개의 요인을 추출하여 만든 선형 회귀모델이다. 그러나 이 모델은 선형모델의 대표적인 것이긴 하지만, 실제로는 비용에 영향을 미치는 많은 비선형 요인을 고려하지 못함으로써 정확한 비용을 산출하지 못하고 있다.

2) Farr & Zagorski(1965)

프로그램 복잡도를 가장 중요한 비용요인으로 취급한 선형모델을 개발했다.그러나 실제 비용에 영향을 미치는 많은 비선형 요인이 있기 때문에 정확한 비용산출이 어렵다.

3) Aron(1969)

프로젝트의 난이도와 기간에 따른 생산성을 계산한 표를 만들었다. 따라서 아주 간단한 방법으로 개발 EFFORTS를 구할수 있는데, 난이도와 개발기간의 두가지 요인만 고려하기 때문에 부정확한 모델이 되기 쉽다.

4) TRW(Wolverton, 1974)

한 DSI(Delivered Source Code Instruction) 당 가격을 0-100사이의 난이도, 프로젝트 타입, 응용 어플리케이션의 신구성 등의 관계를 함수로 나타내었으며, 소프트웨어를 단위별로 나누어서 평가하는데 유용하다.

5) SLIM(Putnam, 1978)

Raleigh분산을 이용하여 Putnam의 분석에 기초를 둔 모델로써 현재 많이 사용되며 입출력의 관계를 다양하게 분석할 수 있는 모델이다.

6) Doty(Head, 1977)

SDC모델의 데이터를 이용한 확장모델로써 DSI가 10만 이상과 이하로 나누어 두개의 함수로 주어진다.

7) RCA PRICE S(Freiman, 1979)

가격 변이치를 복잡도와 요소에 관계되는 함수로써 표현할 수 있으며 단위별로도 분석이 가능하다.

8) IBM-FSD(Walston, Felix, 1977)

29개의 특성을 이용하여 가중된 합으로 생산성을 계산할 수 있다.

9) Boeing(Black, 1977)

5-6개의 특성을 이용하여 크기를 변수로 주어 Man-Month를 측정하는 모델이다.

10) GRC(Carriere, Thibodeau, 1979)

매우 다양한 상관관계를 측정할 수 있는 모델이다.

11) Bailey-Basili Meta(Bailey, Basili, 1981)

통계적인 방법을 이용하여 상당히 정확한 추정치를 계산할 수 있는 모델로써 프로그램의 규모와 방법, 복잡도, 개인의 경험도 등을 고려하여 $E=a(SIZE)^b+c$ 이라는 함수식으로 나타냈다.

12) COCOMO(Boehm, 1981)

SDLC의 진행에 따라 3개의 모델이 있으며 주어진 프로젝트의 유형에 따라 다시 3개의 관계함수가 주어지고 15개의 특성치가 고려되어 비교적 정확한 예측력을 갖는 모델이다. 그러나 비용요인을 객관화 시키는데 어려움이 있다.

〈표 2〉 모델의 함수적 특성
〈Table 2〉 Functional characteristics of algorithm model

모 델	함수적 특성
SDC	Linear Model
IBM—FSD,DOty	Multiplicative Model
Halstead,Putnam	Analytic Model
Aron,TRW,BOEING	Tabular Model
RCA PRICE S,SLIM,COCOMO	Composite Model

이 외에도 다음과 같은 모델이 있다.
- SOFTCOST 모델(Tausworthe, 1981)
- COPMO 모델(Thebaut, 1983)
- Jensen 모델(1984)

2.3 통계적 접근방법

1) 베이지안 기법

이 기법은 과거 유사 프로젝트 자료와 전문가의 의견을 결합하여 소프트웨어의 규모를 산정하는 방법이다. 소프트웨어 개발을 위한 초기 계획 단계에서 요구분석이 완전히 이루어지지 않았을 때에 통계학의 베이지안 규칙을 이용하여 소프트웨어의 규모를 산정하는 것이다.

2) 회귀분석기법

이 기법은 회귀식에 의해 규모를 산정하므로 회귀식의 유도를 위하여 충분한 과거의 데이터로부터 소프트웨어의 규모에 직접적으로 영향을 미치는 변수를 추출하여야 한다.

회귀분석 결과에 의하면 비록 변수 선택에 수정이 요구되기는 하지만 비선형 모델의 필요성에 대해서는 신중한 분석이 요구된다. 모델에 대해 복잡한 비선형방정식을 적용한다는 것은 비실용적이므로 선형모델로 부터 산정된 값을 수정하는 것이 바람직하다.

Itakura[21]는 프로그램의 규모에 영향을 미치는 요인들을 구조적으로 분석하고 11개의 변수를 사용하여 다중회귀분석에 의한 추정모델을 제안하여 특정 업무의 프로그램 규모를 예측하는데 사용하였다.

2.4 Case-Based Reasoning에 의한 접근 방법

S/W개발 공수의 예측시 과거에 수행한 적이 없는 프로젝트에 대한 데이터가 존재하지 않는 상황하에서는 일반적으로 예측이 곤란하거나 개략적인 추정밖에 할 수가 없다. 그러나 CBR에 의한 접근방법은 과거의 경험이 있는 개발자나 또는 유사한 프로젝트의 사례를 찾아서 이들과 상황이 비슷한 입출력 변수를 매핑시켜 유사한 예측치를 찾아냄으로써 이러한 문제를 해결한다. 이를 위해서는 프로젝트의 특성과 프로그램의 규모,프로그래머의 능력등의 분류체계를 수립하여 특정도메인(Domain Specific)의 지식을 데이터베이스화 한다. 그리고 이들을 규칙베이스(Rule Base)를 통해 추론을 수행해서 관련 사항을 추출하여 S/W개발 공수의 예측을 수행하면 보다 효율적인 모델이 된다.

Mukhopachyay[4]는 CBR을 이용하여 정확한 S/W개발 공수를 추정하는 방법을 제안했다. 이러한 기법을 COCOMO와 Function Point와 대비시켜 보다 예측력이 좋은 결과를 제시했다.

2.5 인공신경망 추정모델

기존의 모델에서는 예측요소의 선정에 따라서 그 결과가 상당한 차이가 나타나는데 이러한 요소의 선정과 이들의 가중치값을 결정하기가 용이하지 않아 이를 사용할 때에는 신중성이 요구된다. 신경망 모델에서는 비교적 이들을 선정하고 그 값을 결정하는데 융통성이 많고 자유롭게 요소의 추가·삭제·수정이 용이하다. 또한 신경회로망의 특징은 첫째, 데이터들로부터 포착하기 어렵고 알려지지 않은 관계를 유추해낼 수 있고,둘째, 학습데이터와 비슷한 유형에 대해 정확하게 반응하여 일반화 능력을 갖고 있으며 셋째, 비선형적 문제를 다루기 때문에 복잡한 문제에 대해서 그 비선형 특성을 잘 반영할 수 있다.

Venkatachalam[1]은 신경망에 의한 S/W개발공수의 예측모델에 관한 기본 아키텍춰를 개념적으로 제시했다. 그는 22개의 입력 노드와 2개의 출력노드를 갖는 다층 네트워크를 설계하고 백프로퍼게이션(Back propagation) 알고리즘을 적용하는 모델을 제시했는데, 여기서 사용한 입출력노드는 Boehm[18]이 COCOMO 에서 사용한 변수를 동일하게 적용하였다.

- 입력노드의 변수의 정의 : S/W프로젝트의 특성을 세분화해서 각기 구분되는 22개의 요소로 구분했다.
- 출력노드의 변수의 정의 :
  - 개발투입공수로 Man-Month로 산출
  - 프로젝트 종료시 까지 걸리는 개발 총 기간(Month)
- 은닉층 노드정의 : 은닉층의 수와 은닉층에

서의 노드수의 결정은 일반적으로 휴리스틱한 지식에 의해 결정된다.

그러나 구체적인 신경망의 훈련과정과 예측결과를 제시하지 못하고 개념적 토대만을 제시하여 본 연구의 결과와 비교할 수가 없었다.

2.6 기존 예측모델의 한계점

기존의 S/W개발 예측모델들의 공통적인 문제점은 여러 다른 모델로 부터 유도된 COST와 EFFORTS가 실제값보다 상당한 차이를 보이고 있다는 것이다[1]. 이러한 차이는 프로젝트 관리자로 하여금 실제 투입자원의 적정량을 결정하는데 상당한 혼란을 주고 있다.

또한 과거 데이터에 기초한 모델들은 향후 정보화 기술의 발전동향을 반영하지 못하고 있다. 예를 들면 프로그램언어, H/W, S/W공학등의 신기법의 영향인자의 출현에 대해 모델에서 즉시로 반영할 수가 없다는 것이다. 이하에서는 그 동안의 논문에서 기존의 소프트웨어 개발공수의 예측모델의 문제점으로 지적된 내용을 기술한다[4].

- S/W비용산정에 영향을 미치는 요소들의 중요한 특성들을 간과하고 있고, 투입공수의 과소평가로 신뢰성이 낮은 S/W가 생산되고 있다.
- 과대 평가된 S/W개발투입공수로 생산성이 저하되고 있다.
- 분석적 모델에 기초한 계량적 예측모델이 있음에도 불구하고 유효성 검증의 문제가 발생한다.
- 현재의 알고리즘모델은 예측력의 정확도가 부족하여 실제로 사용하는데 한계가 있다.
- 기존의 모델들이 널리 사용되지 못하여 사용상의 편이성, 보편성이 없다.
- 경험 있는 개발자로 부터 좀더 정확한 예측을 할 수 있는 모델이 필요하다.
- 현재의 S/W개발 투입공수 추정모델의 개선에 대한 연구 개발이 필요하다.
- Knowledge-Base접근 방법으로 예측력의 정확도를 증대할 수 있다.

이와 같은 한계점을 극복하기 위해 새로운 접근방법으로 예측모델의 정교성을 향상시키려는 노력이 최근에 이루어 지고 있다.

다음 〈표 3〉은 이들 모델의 접근방법을 요약한 것이다.

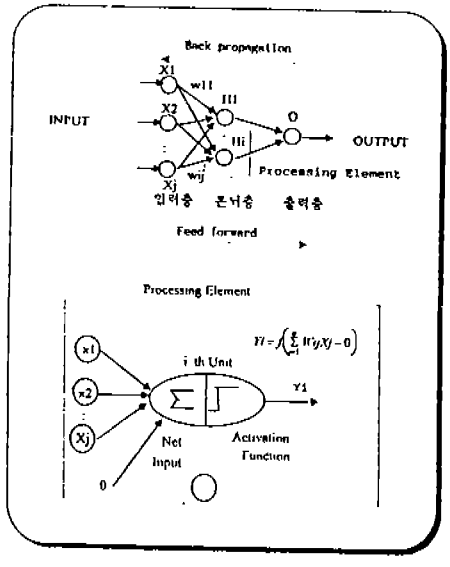
〈표 3〉 예측모델을 위한 다양한 접근방법
〈Table 3〉 Various approaches for estimation

접근방법	제안자	년도
Case Based Reasoning	Mukhopadhyay	1992
〃	Vicinanza	1991
Function Point Analysis	Wrigley	1991
〃	Verner	1989
〃	Symons	1988
Neural Network	Venkatachalam	1993
4th Generation Language	Verner	1988
Anaytic Hierarchy Process	Lee	1993
Metric-Based Classification Trees	Porter	1990

3. 신경망에 의한 예측모델 설계

3.1 신경회로망 모델의 특성

신경망을 훈련시키는 많은 종류의 알고리즘이 있지만 그중 백프로퍼게이션(Back propagation) 패러다임이 예측 및 분류의 문제에 있어서 가장 각광받고 있는 학습 메커니즘이 되고 있다. 백프로퍼게이션 신경망은 여러 층(Layer)으로 구성되며 각 층은 처리단위(Processing Element)와 연결(Connection)로 구성된다.



(그림 1) 신경망 구성도
(Fig. 1) Computational elements of neural networks

첫번째 층은 입력층으로써 입력변수들의 집합으로 표현되는 노드(또는 뉴런)들을 포함한다. 출력층은 출력 변수들의 집합으로 표현되는 노드들을 포함하게 된다.

입력변수와 출력변수 사이의 관계가 비선형(Non-linear)일때 은닉층(Hidden Layer)이 보다 더 고수준의 특성들을 추출해 주며 일반화를 시켜준다. 노드들 사이의 연결은 연결강도의 가중치(Weight)에 의해 조정되는데 이는 훈련셋(Training Set)으로 부터 예(Examples)들을 반복적으로 Feeding시킴으로써 훈련과정에서 조정된다. 이때 각 노드들은 연속 또는 이산 값으로 지정되는 처리수준(Activation Level)을 갖는다. 은닉층 또는 출력층으로 들어오는 노드의 값(Internal Activation)들은 각각의 노드들로 부터 들어오는 처리수준에 상대적인 연결 가중치를 곱하여 그 노드에 들어오는 모든 것을 합하게 된다.

입력층의 노드에서는 처리수준들이 환경으로 부터 입력되는 신호에 따라 결정된다. 이때 노드의 값들은 전이함수(Transfer Function)에 의해 수정되어 출력으로 변환되고 차례로 다음의 노드의 입력으로 된다. 입력 신호를 출력 신호로 변환시키는데 사용되는 전이함수로는 일반적으로 S자형의 Sigmoid함수가 된다. 이러한 Sigmoid함수는

$$F(Y)=\frac{1}{(1+e^{-Y})}\quad Y:\text{Internal Activation}$$

이 된다.

백프로퍼게이션 신경망에서는 모든 연결 가중치들이 출력 오차를 줄이도록 해주고 있다. 여기서 오차는 실제 출력의 값과 신경망에서 추정된 값과의 차이로 이 오차값들은 출력층에서 계산되어 역방향으로 그 이전에 있는 층(Layer)에 전파하여 다시 연결 가중치를 조정하게 하는 메커니즘이다. 따라서 다음과 같은 델타룰(Delta Rule)을 사용한다.

$$\triangle_P W_{ji}=\eta(t_{pj}-o_{pj})i_{pi}=\eta\delta_{pj}i_{pi}$$

.$\triangle_P W_{ji}$: 입력층 i유니트로 부터 출력층 j유니트에의 연결가중치 변화량

.t_{pj} : p번째 목표출력 패턴의 j성분

.o_{pj} : p번째 입력패턴으로 부터 계산된 실제 출력의 j성분

.i_{pi} : p번째 입력패턴의 i성분

.η : 상수

.$\delta_{pj}=t_{pj}-o_{pj}$

$$\triangle_P W_{ji}(n+1)=\eta\delta_{pj}o_{pi}+\alpha\triangle_P W_{ji}(n)$$

.n : 학습횟수

.α : 상수

.$\triangle_P W_{ji}(n)$: 오차 모멘텀

훈련과정은 실제 관찰값들로 부터 입력과 출력 데이터의 Feeding과정을 통해서 오차값들을 역전파시켜 오차의 값들이 사용자가 원하는 임계치(Tolerence level)의 이하가 될 때까지 연결강도를 반복적으로 적용시켜 나가는 과정이 된다. 이와 같은 백프로퍼게이션학습은 신경망 관련 응용에 있어서 가장 많이 이용되는 바, 그 이유는 역전파 학습이 갖는 넓은 응용력과 높은 일반화 능력때문이다. 본 모델에서도 이러한 이유로 역전파 학습을 채택한다.

신경망의 수리적 측면과 기하학적 측면은 두가지의 중요한 특성을 제공한다. 첫째는 견고성(Robustness)으로, 신경망은 일반적으로 오류 극복성(Fault Tolerence)을 갖는데 이는 특정의 몇몇 처리단위(Processing Element)에 오류가 발생하여도 신경망의 전체적인 기능이 크게 영향을 받지 않음을 의미한다.그러므로 신경망의 성과는 오류의 정도가 증가됨에 따라 점차적으로 감소하는 추세를 보임으로써 급격히 변하는 환경이나 예측치 못했던 환경에서 신경망이 안정적으로 기능하는데 크게 기여한다. 둘째는 학습성(Learnability)으로, 신경망은 주어진 학습 입출력 자료로 부터 숨겨진 규칙성을 찾아낼 수 있다. 이러한 규칙성은 신경망내에 분산되어 있는 PE에 각각 저장되며, 이것이 결국 지식베이스내의 지식에 해당된다. 이와같은 신경망의 지식은 수리적 가중치로 표현되고 학습기법에 의해 자동적으로 얻을 수 있다는 것이 큰 장점이 된다.

3.2 사용변수의 정의

S/W를 개발하는데 소요되는 공수를 예측함에

있어서 영향을 미치는 요소는 다양하다. Boehm은 이를 15가지의 요인으로 추출하여 개발 S/W 제품의 특성, 컴퓨터 환경특성, 개발인력 특성, 프로젝트 특성등으로 분류하여 각 비용요소들이 미치는 정도를 수치화 하였다. Fairly는 프로그래머의 능력, S/W의 복잡도, S/W크기, 가용시간, 요구되는 S/W의 신뢰도 및 기술수준등을 주요 요소로 정의하였다.

이외에도 S/W개발에 영향을 미치는 요소로 프로그램언어, 시스템규모, 프로젝트경험도, 복잡성, 프로그램 유형, 문서화, 4세대언어등이 있다.

본 연구에서 신경망모델의 변수설정은 첫째, 실증적인 데이터의 수집가능성을 고려하고, 둘째는 일반적으로 소프트웨어 개발에 영향을 미치는 요소로 Boehm이 제시하는 요인들을 같이 고려하여 입력과 출력 변수로 사용한다.

1) 입력층(Input Layer)노드의 변수

신경망에 입력 신호로 사용될 변수로 출력값에 영향을 미치고 또한 수집 가능한 것이어야 한다. 각 변수의 내용을 정리 하면 다음과 같다.

변수	의 미	값의 형태
X1	정책에 관련된 업무비율	Cardinal
X2	관리에 관련된 업무비율	Cardinal
X3	일상처리에 관련된 업무비율	Cardinal
X4	프로그램본수	Cardinal
X5	업무난이도	Cardinal
X6	개발 업무의 유형	Norminal
X7	개발프로그램의 라인수(KDSI)	Cardinal
X8	S/W개발생산성	Cardinal
X9	사용언어의 종류	Norminal

2) 출력층(Output Layer)노드의 변수

신경망에서 최종 목표로 하는 출력될 변수로서 개발 소요공수로 정의하고 그 단위는 다음과 같다.

- 소요공수(Efforts)의 단위:Man-Month

3) 은닉층(Hidden Layer)노드의 정의

은닉층의 수와 은닉층에서의 노드수의 결정은 일반적으로 휴리스틱한 지식에 의해 결정된다고 할 수 있다. 일반적으로는 다음과 같은 방법으로 중간의 은닉층의 노드 수를 결정한다.

- 휴리스틱한 결정
  - (2n + 1)개, n : 입력 노드의 수
  - (0.1N)개, N : 훈련입력의 수
  - (1.414n + m) to (2n + 1)

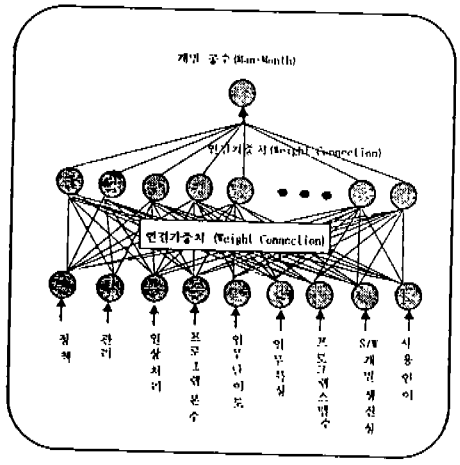
일반적으로 은닉층의 PE의 수는 신경망의 성과에 커다란 영향을 주지 않는다고 한다. 따라서 은닉층의 수와 PE갯수의 결정은 주관적으로 결정되어 신경망 응용에 따라 달라지게 되는데 그 타당성은 실험을 통해 확인하는 수 밖에 없다.

만약 입력자료가 특성 추출이 용이하지 않은 자료로 구성되어 있으면 그러한 입력자료로 부터 고수준의 특성을 추출하기 위해서는 여러개의 은닉층이 요구된다. 반면에 어느정도 고수준의 특성치를 나타내고 있으면 하나 또는 두개 정도의 은닉층만 있어도 거의 모든 형태의 문제해결공간을 구성할 수 있다[16].

그러므로 본 연구에서는 일단 은닉층을 1개로 설정하여 실험을 하기로 한다.그리고 은닉층의 갯수는 신경망의 성과에 그다지 큰 영향을 주지 않는다고 알려져 있기 때문에 은닉층의 노드수는 휴리스틱하게 주어 실험을 한다.

3.3 신경망 모델의 아키텍춰 설계

본 연구에서 제안하는 S/W개발 공수 예측모델은 전방향, 역전파학습 알고리즘을 갖는 신경망으로서 최적해를 갖는 메카니즘으로 설계되어 있다. 따라서 입력 신호는 입력층, 은닉층, 출력층 방향의 전방향연결로 전달되고, 학습은 입력신호의 흐름과는 역방향으로 데이터의 오차를 줄여 가면서 연결 가중치를 계산한다.



(그림 2) 신경망에 의한 S/W 개발공수 예측모델의 구성도
(Fig. 2) Architecture of neural networks

3.4 학습과정

전체 소프트웨어 개발업부 115개중 34개를 테스트 데이터 셑으로 하고 81개 데이터를 학습용 데이터 셑으로 하여 백프로퍼게이션 신경망 모델에 학습을 시켰다. 입력층 노드수는 9개로 하고, 출력층 노드수는 1개 그리고 중간의 은닉층은 1개로 하되, 중간층의 은닉 노드수는 1개 부터 30개까지 설정하여 최적의 중간 노드수를 휴리스틱하게 찾아내는 방법을 사용하였다. 또한 실험을 통하여 학습 싸이클 수(Epoch)도 1000번부터 시작하여 10,000번까지 학습을 시켜 0.001%에서 0.002%범위의 학습 오차제곱총합(Total Sum of Square)을 갖는 데이터 군만을 선정하였다.

이들을 34개의 테스트데이터 셑을 사용하여 테스트시키고 이들 중 MRE(Magnitude of Relative Error)가 50% 이하인 것만을 예측모델 후보로 선정했다. 여기서,

$$MRE=\frac{|MM_{est}-MM_{act}|}{|MM_{act}|}*100,$$

MM_{est}:개발공수(Efforts)의 예측치

MM_{act}:개발공수의 실제치

일반적으로 MRE의 평균 $\overline{MRE}=[\sum_{i=1}^{n}MRE(i)]/n$의 값은 0에 가까울 수록 적합도가 높은 것으로 간주된다[23].

위의 공식에 의해 계산하면 다음 〈표 4〉에서 보는 바와 같이 MRE가 46%이하로 나타났다.

〈표 4〉 은닉층의 노드수와 Epoch수에 따른 MRE
〈Table 4〉 MRE of paired number of epochs and hidden nodes

(단위:%)

은닉층 노드수 (H)	학습 싸이클(Epoch) 수					
	1000	1500	2000	3000	5000	7000
5	45.29	45.18	43.37	43.21	42.50	43.36
10	39.35	39.34	38.92	39.26	38.54	36.70
15	40.94	39.94	39.89	39.31	39.22	36.50
18	39.54	38.37	36.78	34.57	33.62	33.40
25	41.34	40.07	39.69	39.89	39.60	39.69

이 표에서 보면 각 (노드수, Epoch수)별로 최소의 MRE를 갖는 것들은

A:(H= 5,Epoch=5000) => 42.5%
B:(H=10,Epoch=7000) => 36.7%
C:(H=15,Epoch=7000) => 36.5%
D:(H=18,Epoch=7000) => 33.4%
E:(H=25,Epoch=5000) => 39.6%

로 나타났다. 이들을 다시 34개의 테스트 데이터 셑에 대해 다음과 같이 통계적 검정을 해서 가장 예측력이 좋은 것을 선정 하였다.

〈표 5〉 최적 노드수 및 Epoch수의 통계량
〈Table 5〉 Statistics of optimal paired number of epochs and nodes

	Average	STD of Average	MAX	MIN
A(H:5,Epoch:5000)	42.5%	22.8%	90.5%	2%
B(H:10,Epoch:7000)	36.7%	23.3%	80.8%	0.7%
C(H:15,Epoch:7000)	36.5%	23.4%	84.2%	4.55
D(H:18,Epoch:7000)	33.4%	23.5%	88%	0.01%
E(H:25,Epoch:5000)	39.6%	21.6%	85.1%	2.25%

그러므로 소프트웨어 개발공수를 예측하는 신경망모델로 은닉노드수 18개, 학습 Epoch수 7000번인 D(H : 18, Epoch : 7000)가 가장 최적의 예측모델 아키텍춰로 선정되었다.

4. 수정 COCOMO와 다중 회귀분석모델

본 신경망 모델과 비교할 모델로서 기존의 알고리즘 모델중 COCOMO와 통계적 접근방법인 다중회귀모델에 의한 예측값을 구한다.

4.1 수정 COCOMO

Boehm이 제시한 방정식을 K공사의 실정에 맞게 상수와 모드를 조정하기위해 $MM=\alpha(KDSI)^{\beta}*EAF$의 식에서 양변에 log를 취한후 $\log(\alpha)+\beta\log(KDSI)=\log(MM/EAF)$식으로부터 α,β의 값을 최소자승법으로 구하는 식은 다음과 같다.

모 델	내 용
수정 COCOMO	• $Efforts(M.M)=\alpha(KDSI)^{\beta}*EAF$ EAF:개발요소 가중치 • α,β의 값을 실데이터에 적용하여 최소자승법으로 구함

$\log(\alpha)=\frac{a2*d0-a1*d1}{a0*a2-(a1)^2}$, $\beta=\frac{a0*d1-a1*d0}{a0*a2-(a1)^2}$

a0=N:개발업무 수 $a1=\sum\log(KDSI)_i$

$a2=\sum[\log(KDSI)_i]^2$ $d0=\sum\log(MM/EAF)$

$d1=\sum\{\log(MM/EAF)_i*\log(KDSI)_i\}$

그러므로

a0=68, a1=90.1 a2=137.4 d0=92.5

d1=137.5

$MM=\alpha(KDSI)^{\beta}*EAF$

$\log\alpha+\beta*\log(KDSI)=\log(MM/EAF)$

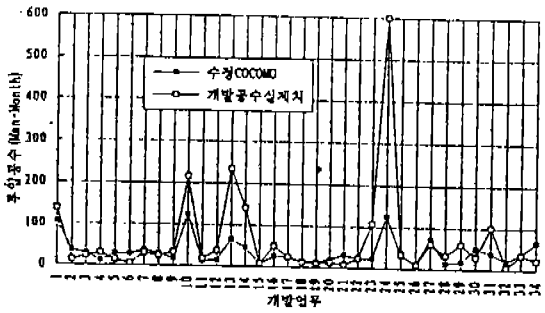
$\log\alpha=(a2*d0-a1*d1)/(a0*a2-a1**2)$

$\beta=(a0*d1-a1*d0)/(a0*a2-a1**2)$

$\alpha=1.82$, $\beta=0.83$이 되고,

구하고자 하는 수정된 COCOMO의 방정식은 $MM=1.82(KDSI)^{0.83}*EAF$가 된다. 여기서 EAF는 Boehm이 제시한 개발비용요소 가중치를 사용했다.

다음 (그림 3)은 34개 데이터의 실제치와 이 모델에 의한 예측결과치의 분포현황을 보여주고 있고, 〈표 6〉은 산출된 34개 데이터의 예측치 통계량이다.



(그림 3) 실제치와 수정COCOMO의 결과치 분포비교
(Fig. 3) Distributions between actual values and revised COCOMO output

〈표 6〉 수정 COCOMO의 예측치 결과와 실제치
〈Table 6〉 The output of revised COCOMO

(단위:Man-Month)

	예측치	실제치
Average	36.6	64.8
std of avg	30.6	110.5
MAX	127.0	600.0
MIN	7.4	5.0

이모델에 의한 예측치와 실제치의 차이에 관한 가설(H0 : $\mu_{ACT}=\mu_{RC}$, H1 : $\mu_{ACT}\neq\mu_{RC}$)을 양측 검정한 결과 유의수준 5%에서 H0가 채택되어 차이가 없는 것으로 나타났다.($\mu_{ACT}=\mu_{RC}$, P-value = 0.15)

4.2 다중 회귀분석모델

통계적 접근방법에 의한 예측모델로 다중회귀분석을 위해 다음과 같이 모델링한다.

1)종속변수(Y) : 9개의 입력변수에 의해 출력되는 투입공수(Man-Month)로 정의

2) 독립변수(X1...X9) : 종속변수에 영향을 미치는 입력변수로 정의

X1:정책관련 비율
X2:관리관련 비율
X3:일상처리 관련 비율
X4:프로그램본수
X5:업무난이도
X6:업무특성
X7:프로그램 스텝수
X8:S/W개발 생산성(월간 1인당 생성코드수)
X9:사용언어 가중치

〈표 7〉 X와 Y의 상관계수
〈Table 7〉 Correlation coefficients of Y and X

사용변수	X1	X2	X3	X4	X5	X6	X7	X8	X9
상관계수	72.6	96.3	159	0.12	-1.9	3.49	1.55	-27.34	4.36

〈표 8〉 회귀분석 결과
〈Table 8〉 The results of regression

결 과	값
constant	-115.0
standard error of Y estimation	54.3
R squared	0.7
degree of freedom	47.0

따라서 회귀분석모델의 방정식은 다음과 같이 도출되었다.

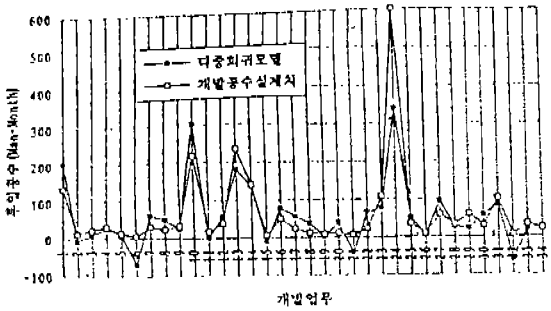
$Y=-115+72.6X_1+96.3X_2+159X_3+0.12X_4-1.9X_5+3.49X_6+1.55X_7-27.34X_8+4.36X_9$

다음 (그림 4)는 34개 데이터의 실제치와 이 모델에 의한 예측결과치의 분포현황을 보여주고

있고, 〈표 9〉는 산출된 34개 데이터의 예측치 통계량이다.

〈표 9〉 다중회귀모델의 예측치 결과와 실제치
〈Table 9〉 The outputs of multiregression output
(단위:Man-Month)

	예측치	실제치
Average	57.9	64.8
STD of avg	88.0	110.5
MAX	339.0	600.0
MIN	−69.0	1.4



(그림 4) 실제치와 다중회귀모델의 결과치 분포비교
(Fig. 4) Distributions between actual values and multiregression output

이모델에 의한 예측치와 실제치의 차이에 관한 가설(H0 : $\mu_{ACT}=\mu_{RC}$, H1 : $\mu_{ACT}\neq\mu_{RC}$)을 양측 검정한 결과 유의수준 5%에서 H0가 채택되어 차이가 없는 것으로 나타났다. ($\mu_{ACT}=\mu_{1R}$, P−value=0.78)

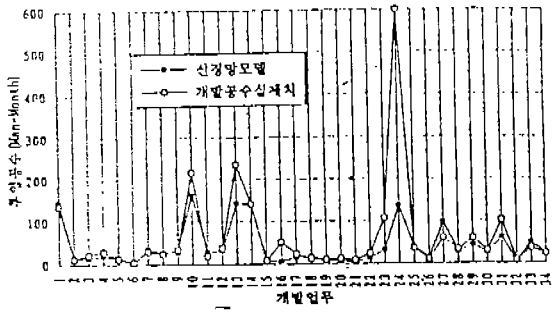
4.3 신경망 모델

다음 (그림 5)는 34개 데이터의 실제치와 이 모델에 의한 예측결과치의 분포현황을 보여주고 있고, 〈표 10〉은 산출된 34개 데이터의 예측치 통계량이다.

이모델에 의한 예측치와 실제치의 차이에 관한 가설(H0 : $\mu_{ACT}=\mu_{RC}$, H1 : $\mu_{ACT}\neq\mu_{RC}$)을 양측 검정한 결과 유의수준 5%에서 H0가 채택되어 차이가 없는 것으로 나타났다.($\mu_{ACT}=\mu_{NN}$, P-value =0.091)

〈표 10〉 신경망 예측치 결과와 실제치
〈Table 10〉 The outputs of neural networks
(단위:Man-Month)

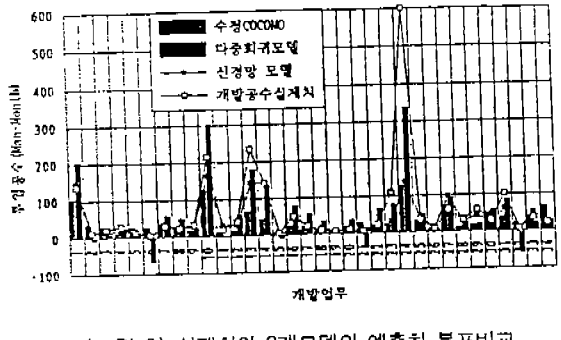
	예측치	실제치
Average	42.3	64.8
STD of avg	47.8	110.5
MAX	164.0	600.0
MIN	1.4	5.0



(그림 5) 실제치와 신경망모델의 결과치 분포비교
(Fig. 5) Distributions between actual values and neural networks

4.4 3개모델의 실제치와 예측치의 차이분석

위 3개의 모델에 의한 예측치와 실제치와의 차이 유무를 통계적으로 검정한 결과 모두 유의수준 5%에서 실제치와의 차이가 없음을 알 수 있었다. 그러므로 위 3개의 모델은 예측모델로 사용을 하여도 큰 무리가 없다고 전제하고 이들 중 가장 예측력이 우수한 모델을 통계적으로 검정하기로 한다. (그림 6)은 실제치와 3개의 모델에 의한 예측치의 분포를 동시에 나타낸 것이다.



(그림 6) 실제치와 3개모델의 예측치 분포비교
(Fig. 6) Distributions of 3 models and actual values

5. 예측결과의 통계적 분석 및 평가

5.1 예측력의 오차 분석

1) 수정 COCOMO의 오차분석

수정된 COCOMO의 공식으로 부터 34개의 테스트 데이터를 입력하여 각각 그 결과값과 실제치의 차이를 구하고 다시 실제치로 나누어 이를

절대값으로 환원한후 백분율(%)을 취한값을 MRE_{RC}로 한다.

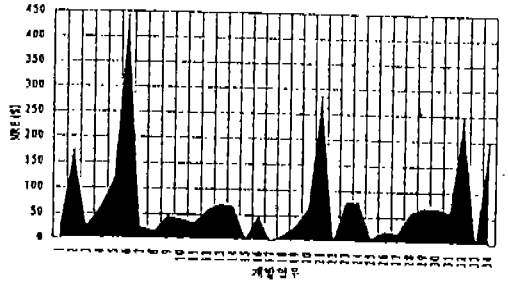
$$MRE_{RC}=\frac{|MM_{est}-MM_{act}|}{|MM_{act}|}*100$$

MM_{est} : 개발공수(Efforts)의 예측치
MM_{act} : 개발공수의 실제치

〈표 11〉은 이렇게 계산된 수정된 COCOMO의 예측치의 오차에 대한 통계량을 구한것이고, (그림 7)은 34개데이터에 대한 예측력의 오차분포를 나타낸 것이다.

〈표 11〉 수정 COCOMO의 MRE
〈Table 11〉 MRE of revised COCOMO

AVG of MRE	STD of MRE	MAX MRE	MIN MRE
76.6%	92.3%	444%	0.42%



(그림 7) 수정 COCOMO의 오차분포도
(Fig. 7) MRE distribution of revised COCOMO

2) 다중 회귀모델의 오차분석

다중회귀모델의 공식으로 부터 34개의 테스트 데이터를 입력하여 각각 그 결과치와 실제치의 차이를 구하고 다시 실제치로 나누어 이를 절대값으로 환원한후 백분율(%)을 취한값을 MRE_{MR}로 한다.

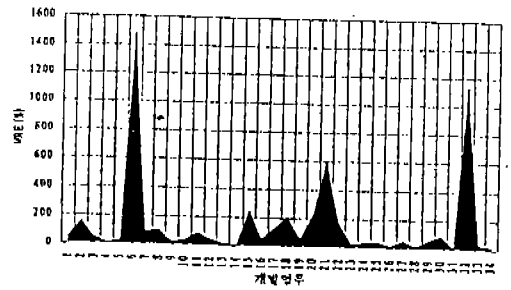
$$MRE_{MR}=\frac{|MM_{est}-MM_{act}|}{|MM_{act}|}*100$$

MM_{est} : 개발공수(Efforts)의 예측치
MM_{act} : 개발공수의 실제치

〈표 12〉는 위식에 의해 계산된 다중회귀모델의 예측치의 오차에 대한 통계량을 구한 것이고, (그림 8)은 34개데이터에 대한 예측력의 오차분포를 나타낸 것이다. 이모델의 오차분포를 보면 실제치의 특정값에는 다른 모델들 보다 근사치로 접근하여 정확한 예측이 되었으나, 전체적으로 보면 (-)값등의 분산이 너무 커서 평균오차가 크게 나타나고 있다.

〈표 12〉 다중회귀모델의 MRE
〈Table 12〉 MRE of multiregression

AVG of MRE	STD of MRE	MAX MRE	MIN MRE
163%	310%	1485%	4.8%



(그림 8) 다중회귀모델의 오차분포도
(Fig. 8) MRE distribution of multiregression

3) 신경망모델의 오차분석

신경망모델로 부터 34개의 테스트 데이터를 입력하여 각각 그 결과치와 실제치의 차이를 구하고 다시 실제치로 나누어 이를 절대값으로 환원한후 백분율(%)을 취한값을 MRE_{NN}으로 한다.

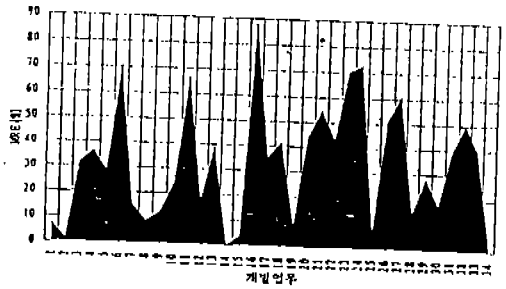
$$MRE_{NN}=\frac{|MM_{est}-MM_{act}|}{|MM_{act}|}*100$$

MM_{est} : 개발공수(Efforts)의 예측치
MM_{act} : 개발공수의 실제치

〈표 13〉은 신경망모델의 예측치의 오차에 대한 통계량을 구한것이고, (그림 9)는 34개데이터에 대한 예측력의 오차분포를 나타낸 것이다.

〈표 13〉 신경망 모델의 MRE
〈Table 13〉 MRE of neural networks

AVG of MRE	STD of MRE	MAX MRE	MIN MRE
33.4%	23.5%	88%	0.01%



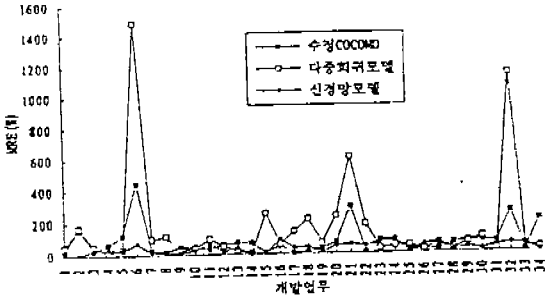
(그림 9) 신경망모델의 오차분포도
(Fig. 9) MRE distribution of neural networks

4) 3개모델의 오차비교 분석

이상의 3개모델에 관한 MRE의 상대적인 비교를 통해 각 모델의 예측오차 정도를 알아본다. (표 14)에서는 3개 모델간의 MRE에 관한 통계량이고 (그림 10)은 34개의 업무별 오차의 차이를 3개의 모델로 동시에 나타낸 것이다.

(표 14) 3개 모델간의 MRE비교
(Table 14) Comparisions of MRE among 3 models

	수정 COCOMO	다중회귀 분석모델	신경망 모델
AVG of MRE	76.6%	163.0%	33.4%
STD of MRE	92.3%	310.0%	23.5%
MAX MRE	444.0%	1485.0%	88.0%
MIX MRE	0.42%	4.8%	0.01%



(그림 10) 3개모델의 오차비교
(Fig. 10) MRE comparisions of 3 models

5.2 모델간의 예측력 비교평가

3개의 모델의 예측력을 비교 평가하기 위해 각 모델의 예측력 오차에 관한 통계량을 중심으로 각각 3개의 가설을 설정하고, 이들간의 상대적 예측력을 비교평가 했다.

1) 수정COCOMO 대 신경망모델의 예측력검정

- 귀무가설(H0) : $\mu_{RC}=\mu_{NN}$
- 대립가설(H1) : $\mu_{RC}>\mu_{NN}$

$\mu_{RC}=\overline{MRE}_{RC}$, $\mu_{NN}=\overline{MRE}_{NN}$

(μ_{RC} : 수정COCOMO의 오차 평균, μ_{NN} : 신경망 모델의 오차 평균)

(표 15) 수정 COCOMO와 신경망의 t-test 검정
(Table 15) Paired t-test comparision between NN and RC

Statistics item	Significance Level	Decision
P-value:0.0041	0.5%	H0 기각

따라서 신경망모델에 의한 예측결과값 오차가 수정 COCOMO에 의한 예측결과 값 보다 작아 신경망 모델의 예측력이 우수하다고 할수 있다.

2) 다중회귀모델 대 신경망모델의 예측력검정

- 귀무가설(H0) : $\mu_{MR}=\mu_{NN}$
- 대립가설(H1) : $\mu_{MR}>\mu_{NN}$

$\mu_{MR}=\overline{MRE}_{MR}$, $\mu_{NN}=\overline{MRE}_{NN}$

(μ_{MR} : 다중회귀모델의 오차 평균, μ_{NN} : 신경망모델의 오차 평균)

(표 16) 다중회귀 모델과 신경망 모델의 t-test검정
(Table 16) Paired t-test comparision between NN and MR

Statistics item	Significance Level	Decision
P-value:0.0075	1%	H0 기각

따라서 신경망 모델에 의한 예측결과값 오차가 다중회귀분석에 의한 예측결과값 보다 작아 신경망 모델의 예측력이 우수하다고 할수 있다.

3) 다중회귀 모델 대 수정COCOMO의 예측력 검정

- 귀무가설(H0) : $\mu_{MR}=\mu_{RC}$
- 대립가설(H1) : $\mu_{MR}>\mu_{RC}$

$\mu_{MR}=\overline{MRE}_{MR}$, $\mu_{RC}=\overline{MRE}_{RC}$

(μ_{MR} : 다중회귀모델의 오차 평균, μ_{RC} : 수정COCOMO의 오차 평균)

(표 17) 수정 COCOMO와 다중회귀 모델의 t-test검정
(Table 17) Paired t-test comparision between RC and MR

Statistics item	Significance Level	Decision
P-value:0.0604	10%	H0 기각

따라서 수정COCOMO모델에 의한 예측결과의 오차가 다중회귀모델에 의한 예측결과 오차보다 작아 이 두개의 모델의 비교에서는 유의수준 10%에서 수정COCOMO모델의 예측력이 우수하다고 할수 있다.

이상의 가설검정에서 보면 3개의 모델중 가장 예측력이 좋은 모델로는 신경망 예측모델이 검증되었으며, 나머지 두개의 모델에서는 수정된 COCOMO가 다중회귀모델 보다 좋은 것으로 나타났다.

6. 결 론

본 연구에서는 효율적인 S/W개발 예측 모델로써 신경망에 의한 예측모델을 제안했다. 제안한 신경망 모델의 예측력을 검증하기 위해서 기존의 알고리즘 예측모델중 가장 합리적인 모델로 인정 받고 있는 COCOMO를 81개의 데이터로 최소자승법에 의해 보정계수를 조정하고, 34개의 데이터에 대해 예측값을 산출하였다. 또한 소프트웨어 개발공수에 영향을 미치는 9개의 독립변수를 선정하여 다중회귀모델을 설계하고 34개의 데이터에 대해 예측치를 산출하였다.

제안된 신경망 모델도 Back propagation 알고리즘 네트워크에 의해 81개의 데이터 셋을 가지고 학습시킨후 34개의 데이터에 대한 결과값을 산출시켜서 위의 2개의 모델과 예측력 비교를 하였다.

이들 3개의 모델에 대한 예측력의 MRE를 비교한 결과 본 연구에서 제안한 신경망 모델에 의한 예측력이 가장 우수하였다.

그러나 본 연구의 신경망 모델의 예측력에 있어서 다소 오차가 크게 나타나고 있는데 이는 학습과정에서 전체데이터의 노이즈가 포함되어 있기때문이다. 이들을 제거하기 위해서는 데이터의 전처리과정(Preprocessing)을 통해서 데이터의 Filtering이 필요하다.

참 고 문 헌

[1] A.R. Venkatachalam, "Software Cost Estimation Using Artificial Neural Networks," in Proceedings of 1993 International Joint Conference on Neural Networks, pp. 987-990, Jul. 1993

[2] T.K.AbdelHamid, "Adapting,Correcting, and Perfecting Software Estimates:A Maintenance Metaphor," IEEE Computer,Vol.10, No.1, pp. 20-29, Mar. 1993

[3] J.Verner and G.Tate, "Estimating Size and Effort in Fourth-Generation Development," IEEE Software, Vol.5, No.4, pp. 15-22, Jul.1988

[4] T.Mukhopadhyay, S.S.Vicinanza, and M. J.Prietuela, "Examining the Fesibility of a Case-Based Reasoning Model for Software Effort Estimation," MIS Quarterly, Vol.16, No.2, pp. 155-171, Jun. 1992

[5] C.D. Wrigley and A.S.Dexter, "A Model for Measuring Information System Size," MIS Quarterly, Vol.15, No.2, pp.245-257, Jun.1991

[6] I.Benbasat and I.Vessey, "Programmer and Analyst Time/Cost Estimation," MIS Quarterly, Vol.4, No.2, pp.31-43, Jun.1980

[7] C.R.Symons, "Function Point Analysis: Difficulites and Impro-vements," IEEE Trans.on Soft.Eng, Vol.14, No.1, pp.2-11, Jan.1989

[8] J.M.Verner and G.Tate,B.Jackson and R.G.Hayward, "Technology Dependency in Function Point Analysis:A Case Study and Criti-cal Review," Commun. of the ACM, Vol.32, No.5, pp. 375-382, May.1989

[9] R.K.Lind and K.Vairavan, "An Experimental Investigation of Soft-ware Metrics and Their Relationship to Software Development," IEEE Trans.on Soft. Eng., Vol.15, No.5, pp. 649-653, May1989

[10] A.J.Albrecht and J.E.Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction:A Software Science Validation," IEEE Trans. on Soft.Eng., Vol.SE-9, No.6, pp. 639-648, Nov.1983

[11] A.A.Porter and R.W.Selby, "Empirically Guided Software Development Using Metric-Based Classification Trees," IEEE Software, Vol.7, No.1, pp.46-54, Mar.1990

[12] S.Henry & C.Selig, "Predicting Source-

Code Complexity at the Design Stage." IEEE Software,Vol.7, No.1, pp. 36-44. Mar.1990

[13] A.L.Lederer, R.Mirani, B.S.Neo, C.Pollard and J.Prasad, "Infor-mation System Cost Estimating:A Management Perspective", MIS Quarterly,Vol.14, No. 2, pp. 159-176, Jun.1990

[14] C.F.Kermerer, "An Empirical Validation of Software Cost Estima-tion Models," Commun.of the ACM, Vol.30, No.5, pp. 416-429, May. 1987

[15] B.W.Boehm, "Understanding and Controlling Software Costs", IEEE Trans. on Soft. Eng., Vol.14, No.10, pp. 1462-1477, Oct. 1988

[16] R.P.Lipmann, "An Introduction to Computing With Neural Nets", IEEE ASSP Magazine, Vol.3, No.4, pp. 4-22, 1988

[17] C.F.Kermerer, "Software Cost Estimation Models," Chapter 25 in Software Engineers Reference Handbook, Butterworth, Surrey, U.K., 1991

[18] B.W.Boehm,Software Engineering Economics,Prentice-Hall, 1981

[19] T.Demacro, Controlling Software Projects, Yourdon Press, 1982

[20] B.Londexi,Cost Estimation for Software Development, Addison-Wesley, 1987

[21] M.Itakura and A.Takayanagi, "A Model For Estimating Program Size and Its Evaluation," in Proc.of 6th International Conference on Soft. Eng., pp. 104-109, 1982

전 응 섭

1992년 한국과학기술원 정보공학과 박사과정
1985년~89년 한국과학기술원 시스템공학센터 연구원
1989년~91년 HP CIM Div. Application Engineer & Consultant
1991년~현재 인덕전문대학 사무자동화과 조교수
관심분야 : 지능형 Workflow System, 전문가시스템, 신경망이론, 인공지능 응용분야, 소프트웨어공학