신경망을 이용한 소프트웨어 개발노력 추정

이 상 운†

요 약

소프트웨어공학에서 소프트웨어 측정분야는 30년 이상 수많은 연구가 있어 왔으나 아직까지 구체적인 소프트웨어 비용추정 모델이 없는 실정이다. 만약 소프트웨어 비용-개발노력을 측정하려면 소프트웨어 규모를 추정해야 한다. 많은 소프트웨어 척도가 개발되었지만 가장 일반적인 척도가 LOC(line of code)와 FPA(Function Point Analysis)이다. FPA는 소프트웨어 규모를 측정하는데 LOC를 사용할 때의 단점을 극복할 수 있는 기법이다. 본 논문은 FP와 기능 구성요소 형태들로 측정된 소프트웨어 규모로 소프트웨어 개발노력을 추정하는 신경망 모델을 제안한다. 24개 소프트웨어 개발 프로젝트 사례연구를 통해 적합한 신경망 모델을 제시하였다. 또한, 회귀분석 모델과 신경망 모델을 비교하여 신경망 모델의 추정 정확성이 보다 좋음을 보였다.

Software Development Effort Estimation Using Neural Network Model

Sang-Un Lee†

ABSTRACT

Area of software measurement in software engineering is active more than thirty years. There is a huge collection of researches but still no a concrete software cost estimation model. If we want to measure the cost-effort of a software project, we need to estimate the size of the software. A number of software metrics are identified in the literature ; the most frequently cited measures are LOC (line of code) and FPA (function point analysis). The FPA approach has features that overcome the major problems with using LOC as a measure of system size. This paper presents an neural networks (NN) models that related software development effort to software size measured in FPs and function element types. The research describes appropriate NN modeling in the context of a case study for 24 software development projects. Also, this paper compared the NN model with a regression analysis model and found the NN model has better estimative accuracy.

키워드 : 소프트웨어 개발노력 추정(Software Effort Estimation), 소프트웨어 규모(Software Sizing), 회귀분석(Regression Analysis), 신경망(Neural Network), 기능점수(Function Point), 전향망(FeedForward Network)

1. 서 론

소프트웨어 개발시 중요하게 제기되는 문제점으로 개발 생명주기의 초기단계에서 개발과 관련된 비용을 추정하는 능력이다. 소프트웨어 측정분야는 30년 이상 수많은 연구가 있어왔으나 소프트웨어 개발비용에 영향을 미치는 다양한 속성들과 이들간의 관계 불명확으로 아직까지 구체적인 소프트웨어 비용측정 모델이 없는 실정이다. 만약 소프트웨어 비용-개발노력을 측정하려면 소프트웨어의 규모를 추정해야 한다. 길이로서 소프트웨어의 규모를 측정하기 위한 최초의 소프트웨어 척도 중 하나가 LOC(Line Of Code)이다[1, 2]. LOC 척도를 기초로 한 모델로는 Boehm[3, 4]의 COCOMO (COnstructive COst MOdel) 모델이 널리 사용되고 있다. 소프트웨어 규모 측정 단위로 LOC를 사용할 경우 LOC에 대한 일반적으로 받아들일 수 있는 정확한 정의 부족, 언어에 종속, 요구분석 또는 설계단계에서 정확한 LOC의 추정이 어려우며 코딩이 종료된 후 측정될 수 있으며, 규모에 대한 특정한 하나의 관점인 길이로서만 기능성 또는 복잡도(Complexity)를 결정하는 문제점이 있다.

이의 대안으로 프로젝트의 규모를 측정하는 척도로서 복잡도와 기능성에 대한 광범위한 연구가 이루어졌다. 복잡도를 측정하는 척도로는 Halsted's Software Science와 McCabe's Cyclomatic Number가 있으며 개발 초기에 계산될 수도 있지만 대부분의 복잡도 척도가 코드로 제한된다.

기능성 척도로는 시스템의 기능성으로 소프트웨어 규모를 측정하는 FPA(Function Point Analysis)[5, 6]가 있다. FPA는 개발과정 초기에 프로그램의 기능적인 측면에서 소프트웨어 생산성을 측정하기 위한 척도로 IBM에 근무하는 Albrecht가 1979년에 제안하였다. 과거 20년 이상 기능적 규모 척도가 소프트웨어공학 분야에서 연구되었으며, Demarco's Bang Metrics[7]에 대한 많은 연구를 수행하였으나 상업적으로 널리 사용되지 못하였는데 비해 FPA는 특정 분야에서 널리 사용되고 있다. FPA는 사용자에게 양도될 시스템의 기능으로 소프트웨어 시스템의 규모와 복잡도를 정량화하는 방

† 정 회 원 : 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당
논문접수 : 2001년 3월 16일, 심사완료 : 2001년 5월 9일

법으로 소프트웨어 프로젝트를 개발하기 위해 사용되는 언어 또는 도구와 독립적, 개발 생명주기의 초기단계인 요구분석 단계에서 측정 가능 등 장점이 있으며, LOC를 사용할 때의 주요 문제점을 극복할 수 있는 접근법이다[2].

기존의 FPA 기법을 이용해 소프트웨어 개발노력(Effort)을 추정하기 위한 모델들 [2,8,9]은 소프트웨어의 정보영역과 복잡도의 주관적인 평가의 계수적 측정을 통한 실험적 관계를 통해 얻어지는 알고리즘적 기법으로 선형회귀 또는 다중회귀 분석을 이용하여, 개발노력이 기능점수(Function Point)나 기능 구성요소 형태(Function Element Type)들에 대해 선형이나 비선형 형태를 취함을 연구하였다. 이 모델들은 알고리즘적 기법을 사용함으로써 주관적으로 복잡도 가중치들이 설정되어 계산되고 복잡한 비선형 형태를 적절히 표현하지 못하였다. 따라서, 본 연구는 자체의 학습 능력을 이용해 주어진 데이터들 사이의 복잡한 비선형 관계를 표현할 수 있는 능력을 가진 신경망 모델을 소프트웨어 개발노력 추정 분야에 적용하여 기존의 회귀분석 모델들 보다 정확한 예측 모델을 제시하고자 한다.

제2장에서는 FPA 기법에 관한 관련연구 및 문제점을, 제3장에서는 소프트웨어 개발노력 추정을 위한 신경망 모델을 제시한다. 제4장에서는 제안된 신경망 모델과 기존의 회귀분석 모델의 결과를 비교하여 제안된 모델이 보다 좋은 추정 결과를 얻음을 제시한다.

2. 관련 연구 및 문제점

LOC를 이용한 소프트웨어 비용 추정 모델은 식 (1)의 형태를 취한다.

$$E = A + B \times (KLOC)^{C} \tag{1}$$

식 (1)에서 E는 예측된 개발노력으로 Man-Months로 측정되고, A, B, C는 상수이며, KLOC는 최종 코딩된 소프트웨어의 KLOC(Thousands of Line Of Code) 대한 추정된 수이다. 예로 Bailey-Basili 모델은 $E = 5.5 + 0.73 \times (KLOC)^{1.16}$ 로, Boehm의 Complex 모델은 $E = 2.8 \times (KLOC)^{1.20}$으로 개발노력을 추정하였다. 그러나 소프트웨어 규모 측정 단위로 LOC를 사용할 경우 발생되는 문제점은 서론에서 제기한 바와 같다.

FP를 이용한 비용추정 모델도 LOC의 식 (1)과 유사한 형태를 갖고 있으며, FP는 다음의 5가지 기능 구성요소 형태들을 이용해 계산된다.

- 내부 논리적 화일(응용프로그램 내에서 유지되는 논리적 데이터)
- 외부 인터페이스 화일(다른 응용프로그램에서 유지되거나 응용 프로그램이 참조하는 논리적 데이터)
- 외부 입력(응용 프로그램 및/또는 제어 과정에서 데이터를 유지하는 능력)
- 외부 출력(보고서 및 메시지)
- 외부 조회(단순 데이터 검색에 대한 요청/응답)

이들 유일한 구성요소 형태 각각에 대해 <표 1>의 기능적 복잡도가 결정된다.

〈표 1〉 복잡도 계산

구 분		Data Element Types		
		1 ~ 5	6 ~ 19	20 이상
Referenced File Types	0 ~ 1	Low	Low	Average
	2 ~ 3	Low	Average	High
	4 이상	Average	High	High

이어서, <표 2>의 가변적인 가중치 (W_{ij})와 각 기능 구성요소 형태 i의 복잡도 레벨 j에 대한 값 (Z_{ij})에 대해 식 (2)의 UFP(Unadjusted FP)가 계산된다.

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} W_{ij} Z_{ij} \tag{2}$$

다음으로, k번째 프로젝트에 대한 최종 조절된 FP는 식 (2)에 식 (3)의 TCF(Technical Complexity Factor)를 곱하여 $FP = c_k \sum_{i=1}^{5} \sum_{j=1}^{3} W_{ij} Z_{ij}$로 계산된다.

$$c_k = 0.65 + \frac{\text{sum of factors}}{100} \tag{3}$$

〈표 2〉 가중치 설정

기능 구성요소 형태	기능 복잡도		
	Low	Average	High
외 부 입 력	x3	x4	x6
외 부 출 력	x4	x5	x7
논리적 내부화일	x7	x10	x15
외부 인터페이스 화일	x5	x7	x10
외 부 조 회	x3	x4	x6

TCF는 각 복잡도 요소 중 영향이 없는 것으로부터 많은 영향을 미치는 것까지 영향의 정도를 분류한 14개 요소에 기초한 비율이다.

FP척도는 소프트웨어 프로젝트 추정, 생산성과 품질 평가에서 일관성과 객관성을 제공하는 구체적으로 공인되고 표준화된 척도로 프로젝트 계획 중에 소요 비용과 일정을 추정하는데 유용하게 사용되며, LOC에 비해 소프트웨어의 개발 초기 단계인 요구분석 단계에서 측정이 가능한 장점을 갖고 있다.

FP를 이용한 개발노력 추정에 관한 연구로서, Albrecht et al. [8]은 개발노력 E가 FP에 대해 식 (4)의 선형형태를 취함을 연구하였고, Matson et al. [2]은 개발노력 E가 FP에 대해 식 (5)의 비선형 형태를 취하는 것이 보다 정확한 추정을 할 수 있음을 보였으며, 또한 FP 보다는 기능 구성요소 형태들을 이용해 직접 개발노력을 추정하는 식 (6)의 회귀모델을 제시하였다.

$$E = -13.39 + 0.0545 \cdot FP \tag{4}$$

$$\sqrt{E} = 1.000 + 0.00468 \cdot FP \tag{5}$$

$$E = 3.80 + 0.00119\widehat{IN^2} + 0.00210\widehat{OUT^2} + 0.00608\widehat{INQ^2} + 0.0045\widehat{FILE^2} \tag{6}$$

비용추정 분야의 대부분의 연구는 알고리즘적 비용모델링에 초점을 두고 있다. 이 과정에서 비용은 측정된 출력을 얻기 위해 척도를 가진 입력 또는 비용과 연관된 수학적 공식을 사용해 분석된다. 이 공식은 과거 이력자료 분석으로부터 제시된 형식적인 모델(Formal Model)에 사용되며, 모델의 정확도는 특정 개발환경을 반영하여 가중치 조절을 포함한 모델의 조정(Calibration)으로 향상된다. 일반적으로 이 방법은 추정에 매우 큰 불일치성을 갖고 있다. Kemerer[10]는 예측된 값과 실측값 사이에 85~610%의 추정 편차가 있었으며, 모델의 조정을 거치면, 이들 모델의 성능을 향상시킬 수 있으나 역시 50~100%의 오차를 나타냄을 보였다. 따라서, 좋은 회귀모형 선택은 적정한 양의 데이터뿐만 아니라 주어진 모델 가정의 신중하고 주의 깊은 분석을 요구한다.

본 논문은 알고리즘적 모델의 문제점을 해소하기 위한 대안으로서, 주어진 문제에 대해 모델 스스로 학습할 능력을 보유한 신경망을 이용해 개발노력과 이에 영향을 미치는 변수와의 복잡한 비선형 형태에 적합한 모델을 제시한다.

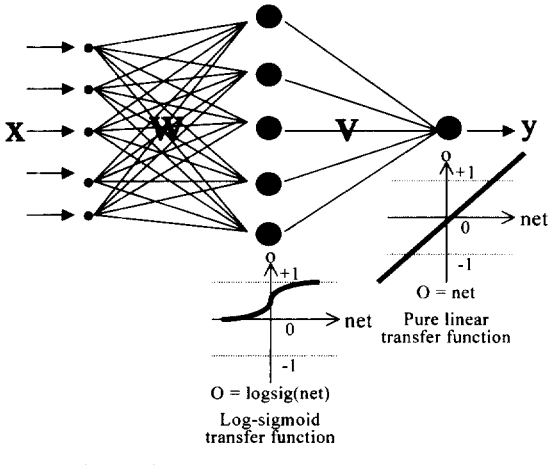
3. 개발노력 추정 신경망 모델

3.1 모델 선택

주어진 문제에 적합한 모델을 선택하는 것이 중요한 사항이다. 사실 많은 다른 신경망 형태와 모델이 적용될 수 있지만, 그 중에서도 가장 많이 알려져 있고, 가장 일반적으로 사용되는 것이 FFN(FeedForward Network)이다. 따라서, 본 논문은 FFN을 이용하고자 한다.

3.2 은닉층 수 및 작동함수 선택

주어진 문제를 표현하기 위한 모델의 적절한 구조(복잡도)를 선택하는 것 또한 문제가 된다. 너무 복잡한 모델은 잡음에까지도 적합되어 과적합이 발생할 수 있으며, 너무단순한 모델은 주어진 문제를 학습할 능력이 없게 된다. 따라서, 주어진 문제를 적절히 표현할 수 있는 모델의 복잡도를 결정하는 것이 중요하다. 본 논문에서는 (그림 1)과 같이



(그림 1) 단일 은닉층을 가진 FFN 신경망

1개의 은닉층을 이용하고 은닉층과 출력층 뉴런의 작동함수로 시그모이드 함수와 선형함수를 각각 사용하고자 한다.

(그림 1)의 모델 구조와 작동함수를 선택한 이유는 다음과 같다.

(1) Cybenko [11]는 비선형 함수인 시그모이드 작동함수를 가진 1개의 은닉층으로 주어진 모든 함수를 표현할 수 있다는 보편적 근사 이론(Universal Approximation Theorem)을 증명해보였으며, Barron [12]은 은닉층에 시그모이드를 출력층에 선형 작동함수를 사용해, 1개의 은닉층을 가진 FFN이 함수근사 능력이 있음을 밝혔다.

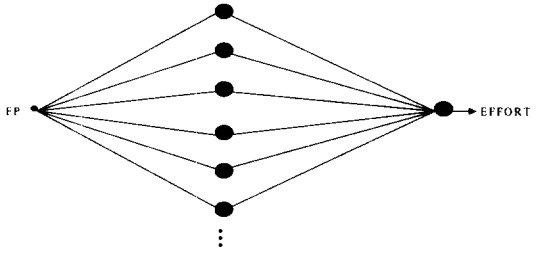
(2) 출력층 뉴런의 작동함수로 시그모이드 함수를 사용하면 이 함수가 표현할 수 있는 한정된 데이터 범위인 [0, 1]로 주어진 데이터를 정규화시켜 훈련을 수행하고 훈련결과 얻어진 데이터를 원래의 데이터 범위로 역변환을 시켜야하는 불편이 따른다. 따라서, 출력층 뉴런의 작동함수를 선형으로 사용하면, 이러한 불편을 해소 가능하다.

3.3 은닉층 뉴런 수 결정

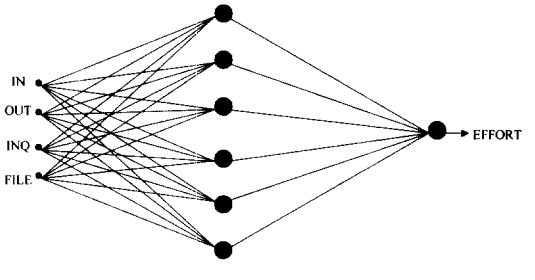
주어진 문제에 적합한 신경망의 은닉층 뉴런의 수를 결정하는 것은 주어진 문제를 적절히 표현할 수 있는 능력으로 문제에 따라 달라질 수 있어 명확한 법칙이 존재하지 않으며, 단지 시행착오법(Trial-and-Error)으로 구할 수 있다. 따라서, 본 연구에는 은닉층 뉴런의 수를 1부터 30개까지 변경시켜 가면서 최적의 결과를 얻는 뉴런 수로 은닉 뉴런 수를 결정한다.

3.4 입-출력 선택

신경망의 입력과 출력 뉴런의 개수를 결정하는 문제는 어떤 데이터를 입력으로 하여 어떤 데이터를 얻을 것인가에 따라 결정된다. 소프트웨어 개발노력을 추정(출력)하기 위해 본



(그림 2) FP 이용 개발노력 추정 신경망 모델



(그림 3) 기능 구성요소형태 이용 개발노력추정 신경망 모델

논문에서는 입력으로 FP만 이용하는 경우 (그림 2)와 기능 구성요소 형태를 이용하는 경우 (그림 3)로 신경망의 입-출력을 결정하고자 한다.

4. 개발노력 추정 실험

Albrecht et al. [6]는 IBM Data Processing Services에서 개발된 24개 소프트웨어 수집 데이터에 대해, 외부 사용자 입력, 조회, 출력과 화일의 수에 의해 프로젝트의 FP 값을 계산하였으며, 입력은 4, 출력은 5, 조회는 4와 화일은 10으로 가중치를 적용하여 $FP_k = (4IN + 5OUT + 4INQ + 10FILE)_{c_k}$를 계산하였다. 특정 프로젝트 복잡도를 ±35% 예로, 20번째 소프트웨어 프로젝트의 FP 값은 $FP_{20} = \{4(69) + 5(112) + 4(21) + 10(39)\} \cdot 1.20 = 1572$로 구해진다. 본 실험에는 <표 3>의 데이터를 이용해 개발노력을 추정하고자 한다.

다른 여러 가지 모델들을 비교하는데 있어, 어떤 의미 있는 척도로서 모델의 추정 정확도를 평가하는 것이 필요하다. 회귀분석의 경우 회귀직선에 의해 종속변수가 설명되는 정도를 나타내는 결정계수(Coefficient of determination) R^2이 있다. 종속변수의 값은 독립변수에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 총 변동을 설명하는데 있어서 회귀직선에 의해 설명되는 변동이 기여하는 비율이 R^2이다. 따라서, $R^2(0 \le R^2 \le 1)$이 0에 가까우면 추정된 회귀직선은 쓸모가 없으며, 값이 클수록 쓸모 있는 회귀직선이 된다. 또한 제안된 모델을 평가함에 있어서 잔차를 분석한다. 잔차(Residual)는 실제 값과 추정된 값과의 차이로, 잔차분석은 단순회귀모형에서 등분산성, 독립성, 정규성, 직선관계의 가정이 옳은가를 검토할 때 가장 많이 사용되는 방법이다. 따라서, 잔차가 어떤 일정한 형태를 취하지 않고 랜덤하게 분산되어 있는 경우가 좋은 모델이 된다. 본 논문에서는 제안된 모델의 성능을 기존 회귀분석 모델들과 비교 평가하기 위해 R^2와 잔차분석과 더불어 상대오차 척도를 사용한다.

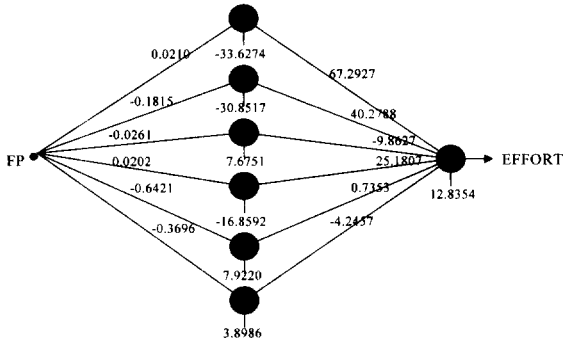
4.1 FP를 이용한 EFFORT 추정

Albrecht et al. [8]은 <표 3>의 데이터에 대해 Effort를 식 (4)와 같이 회귀분석 모델을 제시하였다. 본 연구에서는 은닉층이 1개인 FFN에 대해 은닉층 뉴런의 작동함수로는 시그모이드 함수를, 출력층 작동함수로는 선형 함수를 사용하여, 역전파 알고리즘(Backpropagation algorithm : BP)으로 훈련을 수행하였다. (그림 2)의 신경망을 이용한 실험에서, 은닉 뉴런 수를 1개에서 30개까지 변화시키면서 실험을 수행한 결과 최적의 은닉 뉴런 수(6개)와 가중치는 (그림 4)와 같이 결정되었다. 실험결과 얻어진 개발노력은 (그림 5)에 표기되어 있다. (그림 5)에서 FFN 신경망을 이용한 경우 주어진 데이터에 비선형으로 근사시켜 Albrecht et al. [8]의 선형회귀모델 보다 정확한 개발노력을 추정 할 수 있었다.

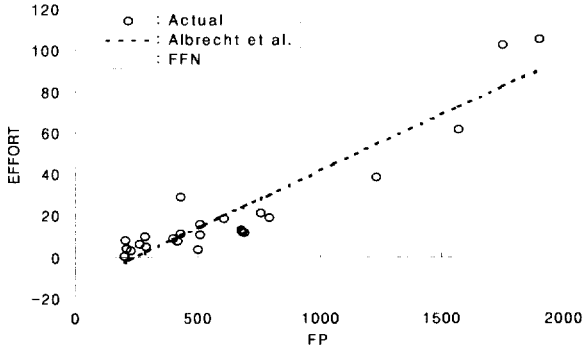
실험결과 얻어진 상대오차와 잔차는 (그림 6)과 (그림 7)

〈표 3〉 IBM 개발노력 데이터

Case No.	기능 구성요소 형태				TCF	FP	개발노력 (M/M)
	입력 (IN)	출력 (OUT)	조회 (INQ)	화일 (FILE)			
1	25	150	75	60	1.00	1750	102.4
2	193	98	70	36	1.00	1902	105.2
3	70	27	0	12	0.80	428	11.1
4	40	60	20	12	1.15	759	21.1
5	10	69	1	9	0.90	431	28.8
6	13	19	0	23	0.75	283	10.0
7	34	14	0	5	0.80	205	8.0
8	17	17	15	5	1.10	289	4.9
9	45	64	14	16	0.95	680	12.9
10	40	60	20	15	1.15	794	19.0
11	41	27	29	5	1.10	512	10.8
12	33	17	8	5	0.75	224	2.9
13	28	41	16	11	0.85	417	7.5
14	43	40	20	35	0.85	682	12.0
15	7	12	13	8	0.95	209	4.1
16	28	38	24	9	1.05	512	15.8
17	42	57	12	5	1.10	606	18.3
18	27	20	24	6	1.10	400	8.9
19	48	66	13	50	1.15	1235	38.1
20	69	112	21	39	1.20	1572	61.2
21	25	28	4	22	1.05	500	3.6
22	61	68	0	11	1.00	694	11.8
23	15	15	6	3	1.05	199	0.5
24	12	15	0	15	0.95	260	6.1



(그림 4) FP를 이용한 개발노력 추정 신경망 가중치



(그림 5) FP를 이용한 개발 노력 추정

에 표기되어 있다. (그림 6)과 (그림 7)에서 FFN이 전반적으로 Albrecht et al. [8]의 선형회귀모형보다 좋은 결과를 나타내었으며, 실험결과 다음과 같은 결론을 얻었다.

(1) Albrecht et al. [8] 모델은 데이터를 모델에 적합시키는데 있어 FP가 1200을 초과하는 수가 매우 작은 수이며

1200을 초과하는 4개 데이터를 제거하면 매우 다르게 적합되는 단점을 갖고 있다. 이 문제를 해결하는 방법으로는 FP와 개발노력 간의 관계가 비선형 형태를 취하도록 하거나 선형관계를 가지면서 FP 수가 증가함에 따라 오차의 분산을 증가시키는 방법이 있다. 본 논문은 첫 번째 해결방법으로 신경망을 이용하여 FP와 개발노력 간의 비선형 관계를 취하여 보다 좋은 모델을 얻었다.

(2) Albrecht et al.[8] 모델은 R^2는 87.4%, MSE(Mean Squared Error)는 97.3219이다. 이에 반해 은닉 뉴런을 6개 사용한 제안된 FFN 모델은 R^2는 93.77%, MSE는 20.7701로 Albrecht et al.[8] 모델보다 향상된 결과를 얻었다.

4.2 기능 구성요소 형태를 이용한 개발노력 추정

FP를 이용해 개발노력을 추정하는 방법에 비해 기능 구성요소 형태를 이용해 개발노력을 추정하는 모델은 FP를계산하기 위한 노력 감소뿐만 아니라 가중치와 TCF 등 주관적으로 설정된 상수 값의 적절성, 기능 구성요소 형태들과 FP 간의 관계 등 복잡한 문제를 해소시킬 수 있을 것이다. Matson et al.[2]는 <표 3>의 데이터에 대해 $c_k\mathrm{IN} = \widehat{\mathrm{IN}}$ 과 같이 치환하여 식 (7)의 회귀모형을 제안하고 이에 대해 식 (6)과 같이 모델을 제시하였다.

$$E = \beta_0 + \beta_1 \widehat{IN} + \beta_2 \widehat{OUT} + \beta_3 \widehat{INQ} + \beta_4 \widehat{FILE} + \varepsilon \qquad (7)$$

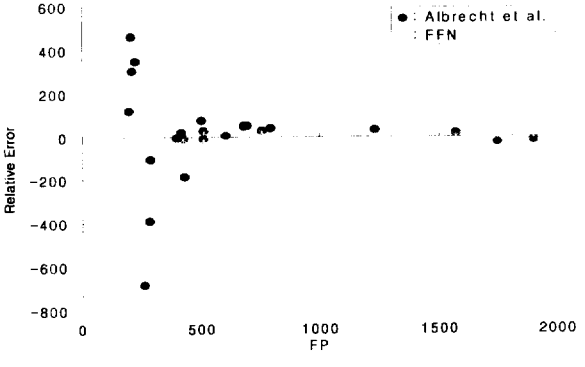
제안된 신경망 모델은 (그림 3)의 신경망을 이용한 실험에서, 은닉 뉴런 수를 1~30개까지 변화시켜 최적의 은닉 뉴런 수를 15개로 결정되었으며, 은닉층의 입력 가중치 $\mathbf{w}(w_{ij}, i =$ 은닉뉴런, $j =$ 입력변수) 은닉층의 출력 가중치 $\mathbf{v}(v_{ij}, i =$ 은닉뉴런, $j = 1$)와 은닉층 뉴런의 바이어스(Bias) $\mathbf{b}_1(b_{ij}, i =$ 은닉뉴런, $j = 1$)과 출력층 뉴런의 Bias b_2는 다음과 같이 얻어졌다.

실험결과 얻어진 개발노력 추정은 (그림 8)에 표기되어있다. (그림 8)에서 제안된 FFN 모델이 Matson et al.[2]의 선형회귀 모델보다 복잡한 비선형 형태를 취해 주어진 데이터에 보다 적합함을 알 수 있다.

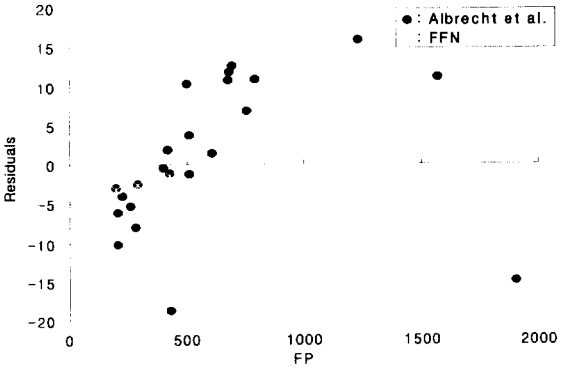
상대오차와 잔차는 (그림 9)와 (그림 10)에 표기되어 있다. (그림 9)에서 FP가 가장 적을 때의 개발노력 상대오차와 (그림 10)에서 FP가 가장 클 경우의 개발노력 잔차가 큰 점을 제외하고는 제안된 FFN 모델이 Matson et al.[2]의 회귀분석 모델보다 추정 결과가 향상되었다. 실험결과 Matson et al.[2] 모델은 R^2는 97.5%, MSE = 25.7을 얻은데 반해은닉 뉴런 수를 15개로 사용한 제안된 FFN 모델은 R^2는 98.6%, MSE는 7.6155로 상대오차나 잔차 분석 결과 Matson et al.[2] 모델 보다 좋은 결과를 나타내었다.

$$\mathbf{W} = \begin{bmatrix} +3.2629 & +4.3719 & -0.1102 & +1.0864 \\ +3.4219 & +1.1120 & +0.1495 & +0.5811 \\ -0.1578 & +1.3539 & +0.5849 & +1.4121 \\ -0.3072 & +3.5466 & +1.8169 & +0.4006 \\ -8.2431 & -0.2974 & +10.9605 & +5.8642 \\ -1.6293 & -1.7149 & -0.7840 & -0.6829 \\ -1.2495 & +5.1410 & -1.8945 & +5.8577 \\ +3.3173 & +3.9096 & +1.2735 & +2.7044 \\ -1.7340 & +3.5076 & -1.2620 & +6.9896 \\ +0.9847 & +1.3571 & +0.3226 & +0.3595 \\ +1.3686 & -11.0161 & -5.0526 & +1.3148 \\ -0.9803 & -0.1672 & +1.1583 & +0.7524 \\ -1.5137 & -2.5742 & -1.5492 & -0.8194 \\ +4.3933 & +2.2476 & +0.1055 & -0.4245 \\ -0.0205 & +0.1623 & +0.1395 & -0.2284 \end{bmatrix}$$

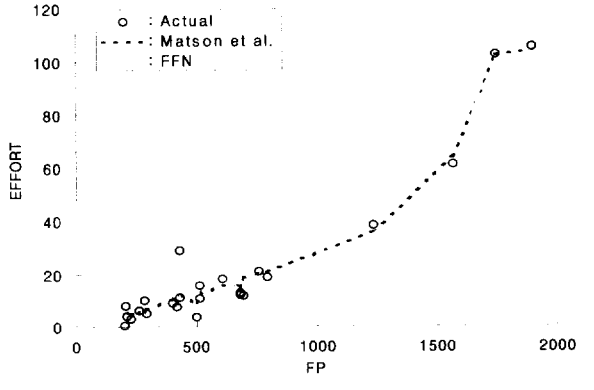
$$\mathbf{V} = \begin{bmatrix} 3.2751 \\ 3.6515 \\ 3.7651 \\ -7.1907 \\ 28.7230 \\ -1.1252 \\ -6.7550 \\ 4.1583 \\ -3.3737 \\ 1.5352 \\ -1.6066 \\ -31.6754 \\ 1.7782 \\ 3.5448 \\ 101.0300 \end{bmatrix} \quad \mathbf{b1} = \begin{bmatrix} 9.1703 \\ 7.5568 \\ 8.0357 \\ -5.9249 \\ -3.7017 \\ 6.7837 \\ 4.0252 \\ -4.3111 \\ 1.5275 \\ 11.1099 \\ 3.9676 \\ 0.2187 \\ -9.4669 \\ 1.7220 \\ -10.5885 \end{bmatrix} \quad \mathrm{b2} = 3.9840$$



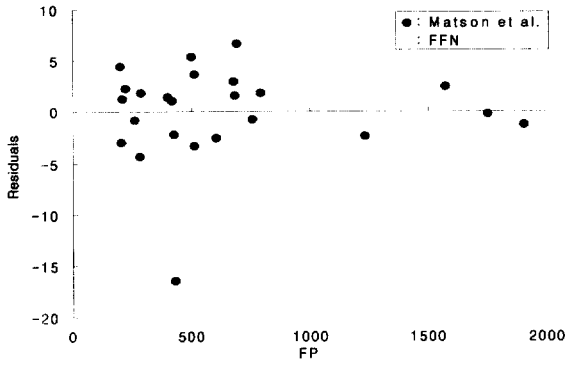
(그림 6) 상대오차 분석(FP vs. Effort)



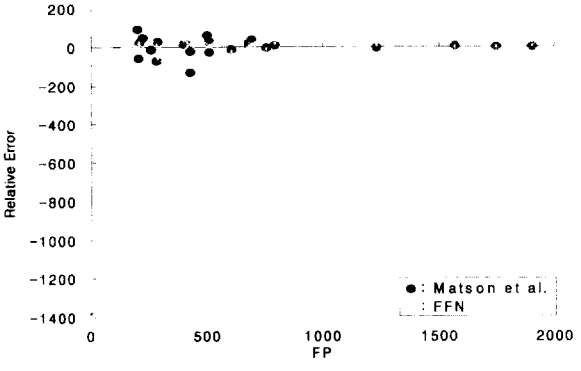
(그림 7) 잔차 분석(FP vs. Effort)



(그림 8) 기능 구성요소 형태를 이용한 개발노력 추정



(그림 9) 상대오차 분석(기능 구성요소 형태 vs. Effort)



(그림 10) 잔차 분석(기능 구성요소 형태 vs. Effort)

5. 결론 및 향후 과제

본 논문은 최근들어 소프트웨어 비용추정 모델 분야에서 일반적으로 사용되고 있는 FP 데이터를 이용, 소프트웨어 개발노력을 추정하는 모델로 신경망 모델을 적용하여 개발노력 추정 능력을 향상시킬 수 있음을 보였다. 기존의 FP를 이용한 개발노력 추정 회귀분석 모델과 FP를 계산하기 위한 소스 데이터인 기능 구성요소 형태를 이용한 개발노력 추정 회귀분석 모델에 비해 제안된 신경망 모델이 종속변수인 개발노력과 독립변수인 FP 또는 기능 구성요소 형태 사이의 복잡한 비선형적 관계를 보다 잘 표현함으로서, 보다 정확한 추정 결과를 얻었다.

신규로 개발될 소프트웨어에 대한 개발노력을 추정하고자 한다면, FP나 기능 구성요소 형태를 계산할 경우, 이들 변수에 대한 값을 기존 개발노력 이력 데이터로 훈련된 제안된 신경망 모델에 입력하면 모델이 갖고있는 가중치 값에 의해 계산된 출력으로 개발노력을 추정 가능하다. 또한 추정된 개발노력을 토대로 FP당 소요비용과 개발기간 등을 알 수 있다면 개발에 소요되는 비용이나 개발기간 추정에도 제안된 신경망 모델을 활용할 수 있을 것이다.

본 연구에는 24개의 소프트웨어 프로젝트에 대한 데이터만을 적용하여 일반화된 모델을 제시하지는 못하였지만 회귀분석 모델보다 정확한 개발노력 추정이 가능한 방법으로 신경망을 적용할 수 있음을 보였다. 따라서, 추후 충분한 개발노력 데이터를 보유한 IFPUG(International Function Point Users Group) 데이터를 활용해 개발노력 추정을 위한 일반적인 신경망 모델 제시와 개발노력에 영향을 미치는 속성에 대한 연구가 수행될 것이다.

참 고 문 헌

[1] L. A. Laranjeira, "Software Size Estimation of Object-Oriented Systems," IEEE Trans. Software Eng., Vol.16, pp.64-71, Jan. 1990.

[2] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software Development Cost Estimation Using Function Points," IEEE Trans. on Software Eng., Vol.20, No.4, pp.275-287.

[3] B. W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

[4] B. W. Boehm, "Software Engineering Economics," IEEE Trans. on Software Eng., Vol.10, No.1, pp.7-19, 1984.

[5] A. J. Albrecht, "Measuring Applications Development Productivity," Proceedings of IBM Application Dev., Joint SHARE/GUIDE Symposium, Monterey, CA, pp.83-92, 1979.

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

[7] T. Demarco, "Controlling Software Projects : Management Measurement & Estimation," New York : Yourdon Press, 1982.

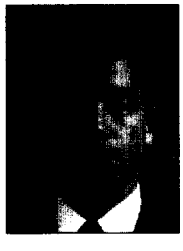
[8] A. J. Albrecht, "Measuring Application Development Productivity," in Programming Productivity : Issues for the Eighties, C. Jones, ed. Washington, DC : IEEE Computer Society Press, 1981.

[9] C. F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," Communication ACM, Vol.30, No.5, pp. 416-429, 1987.

[10] C. F. Keremer, "Reliability of Functional Point Measurement - A Field Experiment," Communications of ACM, Feb. 1993.

[11] G. Cybenko, "Approximation by Super-positions of A Sigmoidal Function," Mathematics of Control, Signals and Systems, Vol.2, pp.303-314, 1989.

[12] A. R. Barron, "Neural Net Approximation," In Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems. New Haven, CT. Yale University, pp.69-72. 1992.



이 상 운

e-mail : sangun_lee@hanmail.net

1983년~1987년 한국항공대학교 항공전자공학과(학사)

1995년~1997년 경상대학교 컴퓨터과학과(석사)

1998년~2001년 경상대학교 컴퓨터과학과(박사)

1992년~현재 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당

관심분야 : 소프트웨어 공학(소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성), 소프트웨어 메트릭스, 신경망, 뉴로-퍼지