훈련데이터 집합을 사용하지 않는 소프트웨어 품질예측 모델

홍 의 석†

요 약

설계 개체의 결함경향성을 판별하는 위험도 예측 모델은 분석이나 설계 같은 소프트웨어 개발 초기 단계에서 시스템의 문제 부분들을 찾아내는데 사용된다. 복잡도 메트릭에 기반한 많은 위험도 예측 모델들이 제안되었지만 그들 대부분은 모델 훈련을 위한 훈련데이터 집합을 필요로 하는 모델들이었다. 하지만 대부분의 개발집단은 훈련데이터 집합을 보유하고 있지 않기 때문에 이들 모델들은 대부분의 개발집단에서 사용될 수 없다는 커다란 문제점이 있었다. 이러한 문제점을 해결하기 위해 본 논문에서는 Kohonen SOM 신경망을 이용하여 훈련데이터 집합을 사용하지 않는 새로운 예측 모델 KSM을 제안한다. 여러 내부 특성들과 모델 사용의 용이성 그리고 모의실험을 통한 예측 정확도 비교를 통해 KSM을 잘 알려진 예측 모델인 역전파 신경망 모델(BPM)과 비교하였으며 그 결과 KSM의 성능이 BPM에 근접하다는 것을 보였다.

A Software Quality Prediction Model Without Training Data Set

Euy Seok Hong†

ABSTRACT

Criticality prediction models that determine whether a design entity is fault-prone or non fault-prone are used for identifying trouble spots of software system in analysis or design phases. Many criticality prediction models for identifying fault-prone modules using complexity metrics have been suggested. But most of them need training data set. Unfortunately very few organizations have their own training data. To solve this problem, this paper builds a new prediction model, KSM, based on Kohonen SOM neural networks. KSM is implemented and compared with a well-known prediction model, BackPropagation neural network Model (BPM), considering internal characteristics, utilization cost and accuracy of prediction. As a result, this paper shows that KSM has comparative performance with BPM.

키워드 : 위험도(Criticality), 예측모델(Prediction Model), KSM

1. 서 론

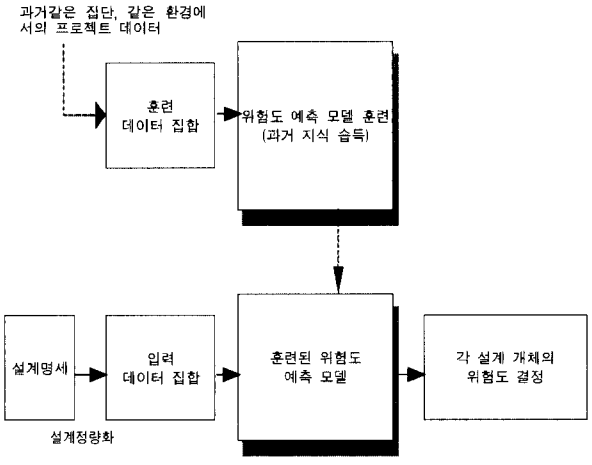
최근에 설계 단계 산물들의 복잡도 메트릭들을 이용하여 최종 개발물의 품질 인자들을 예측하는 소프트웨어 품질예측 모델들에 대한 연구들이 많이 행해졌다. 한 설계 개체의 위험도(criticality)란 개체가 구현되었을 때 갖는 결함경향성(fault-proneness)을 의미한다. 설계 시 위험도 예측은 소규모 시스템 개발에서는 큰 의미를 지니지 못하지만 개발기간이 수년 이상 되는 큰 시스템 개발에서는 시스템 개발 비용을 낮추는 매우 중요한 역할을 한다. 개발 초기 단계에서 문제 부분을 찾아 적절한 자원 할당이나 재설계 결정 등을 할 수 있기 때문이다. 소프트웨어 문제점들의 80%가 20% 정도의 소프트웨어 컴포넌트들에 기인한다는 80 : 20 규칙을 뒷받침하는 많은 실험 연구들이 소프트웨어공학 분야에서 수행되었다[1]. 이는 상대적으로 적은 설계 개체들이 대부분의 소프트웨어 결함들에 영향을 미친다는 것이며 이러한 개체들을 찾아내는 예측 모델들이 소프트웨어 품질 향상에 크게 기여한다는 것을 의미한다. 그러므로 위험도 예측 모델에 관한 연구는 주로 유럽과 미국, 일본의 대형

※ 이 논문은 2001년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2001-003-E00251).
† 종신회원 : 성신여자대학교 컴퓨터정보학부 교수
논문접수 : 2002년 8월 3일, 심사완료 : 2003년 2월 13일

통신 회사들을 주축으로 하여 활발히 진행되고 있다.

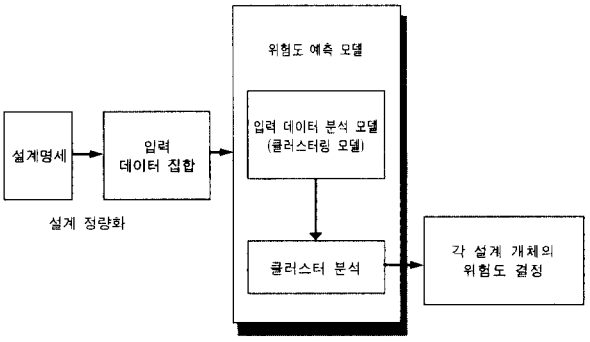
기존에 제안된 위험도 예측 모델들은 많이 알려진 복잡도 메트릭들로 구성된 메트릭 벡터들로 설계 개체들을 정량화 한 후 이들을 위험 그룹과 비위험 그룹으로 분류하는 분류 모델들이 대부분이었다. 분류 모델들은 통계적 기법이외에도 학습 알고리즘이나 패턴 매칭 등 여러 기법들을 사용하였으나 분류 트리 기법을 제외하고는 모델 내부에서 행해진 수행 부분을 역추적하기가 거의 불가능한 블랙박스적인 모델이므로 위험도 결과의 원인을 분석하여 재설계 등의 조치를 취하기가 매우 어려운 문제점이 있다. 이를 해결하기 위해 [2]는 퍼지 분류를 이용하여 원인 분석이 용이한 예측 모델을 제안하였다.

원인 분석의 어려움보다 더 큰 문제점은 앞에서 언급한 모델들 모두가 과거 유사한 개발 환경에서 얻은 실제 프로젝트 데이터인 훈련데이터집합을 필요로 한다는 점이다. 대부분의 개발 집단은 이러한 훈련데이터집합을 보유하고 있지 않으며[3] 설사 과거 데이터를 보유하고 있다 하더라도 모델을 적용하려는 현재 프로젝트의 참여 인력, 개발 환경, 설계 정량화 메트릭 집합이 과거 프로젝트와 매우 유사하여야만 한다는 문제점이 있다. 이러한 문제점은 한 프로젝트를 위해 개발된 모델을 다른 프로젝트에 사용할 수 없다는 편협성의 문제도 동반한다. 그러므로 대부분의 개발 집단은 기존의 예측 모델들을 현재 프로젝트에 직접 적용할 수가 없다. 본 논문은 이 같은 문제들을 해결하기 위해 결과에 대한 원인 분석이 용이하고 훈련데이터집합을 사용하지 않고도 기존 모델들에 근접한 성능을 보이는 예측 모델을 제안한다.



(그림 1) 기존의 위험도 예측 모델 형태

기존의 예측 모델 형태는 (그림 1)과 같다. 훈련데이터집합이란 과거 매우 유사한 개발 프로젝트에서 얻은 오류 데이터(입력 데이터와 해당 오류 정보에 대한 쌍)들의 집합이며 입력데이터 집합은 현재 개발 프로젝트에서 설계 결과를 메트릭 벡터 형태들로 정량화한 데이터 집합이다. 그림에서 모델은 과거 프로젝트에서 발생한 오류 지식을 학습한 모델이며 앞에서 기술한 것과 같이 과거지식을 보유해야하는 것과 과거 지식과 현재 지식이 유사해야한다는 것에 매우 큰 문제점이 있다. (그림 2)는 본 논문에서 제안하는 새로운 예측 모델의 개략적인 형태이다. 제안 모델은 현재 입력 데이터 집합의 분포에 따라 유사한 개체들의 클러스터들을 형성하고 결과 클러스터들을 위험 클러스터와 비위험 클러스터로 나누는 방법을 사용한다. 클러스터링 모델은 무감독형 학습 신경망을 사용하였다.



(그림 2) 새로운 위험도 예측 모델

제안 모델의 입력 대상으로 ITU-T의 표준안으로 널리 사용되고 있는 객체지향 실시간 시스템 명세 언어인 SDL(Specification and Description Language)로 작성한 설계 명세를 사용하며 이를 정량화 하는 메트릭들은 [4]에서 제안한 SDL 메트릭 집합을 사용한다. 이 메트릭 집합의 이론적 타당성 및 유용성은 공리적 검증 방법과 차원 분석(dimensional analysis) 과정을 통해 검증되었다[5].

2장에서는 기존에 제안된 위험도 예측 모델들이 사용한 기법들을 간략히 살펴보고 3장에서는 새로운 예측 모델의 구조와 사용법을 기존 모델들중 좋은 성능을 보인다고 알려진 역전파 신경망 모델과 비교하여 설명한다. 4장에서는 모의 실험을 통하여 역전파 신경망 모델과 제안 모델의 예측 정확도를 비교하고 5장에는 결론과 향후 연구 과제에 대해 기술한다.

2. 위험도 예측 모델

시스템 개발 초기 단계에서 위험도를 예측하기 위한 관련 연구들은 크게 두 가지로 구분할 수 있다. 첫 번째는 과거 유사 프로젝트의 위험도 자료들에 기반한 모델을 만들

어 현재 수행 중인 프로젝트의 위험도 예측에 적용하는 것이다. 대부분의 위험도 예측 모델들은 이러한 형태를 취하지만 앞에서 언급한 많은 문제점들을 지니고 있다. 이러한 예측 모델들의 예로는 판별 분석법[6], 분류 트리법[1], 역전파 신경망 이용 기법[7]이 있으며 이외에도 회귀분석[8], OSR (Optimal Set Reduction)[9], CBR(Case-Based Reasoning) 기법[10], 유전 프로그래밍 기법[11], 유전자 알고리즘에 기반한 퍼지 분류 기법[2] 등이 제안되었다. 분명한 것은 훈련데이터 집합을 요구하는 이러한 모델들이 훈련 데이터와 현재 프로젝트의 데이터의 분포 상에 유사성만 유지된다면 훈련데이터 집합을 필요로 하지 않는 모델보다 매우 훌륭한 성능을 보인다는 것이며 이는 당연하다. 하지만 그런 데이터를 보유한 집단이 거의 없다는 것이 문제이며 본 논문의 목적은 그런 집단에서 사용 가능한 대체 모델을 제안하는 것이다. 앞에서 기술한 여러 가지 모델들의 성능은 데이터의 분포와 프로젝트의 상황에 따라 다른 결과를 내므로 어떤 것이 가장 좋다고 할 수는 없지만 예측 정확도 측면에서만 본다면 역전파 신경망 모델과 CBR 모델이 비교적 좋은 성능을 보이는 것으로 알려져 있다.

두 번째 위험도 예측 관련 연구들은 프로그램의 위험도를 예측할 수 있는 메트릭들을 정의하고 그 타당성을 입증하여 정의한 메트릭들을 바탕으로 시스템의 위험도를 예측하는 연구들이다. 즉 이는 단순히 어떤 설계 개체가 위험한가 아닌가의 여부 결정이 아니라 실제 위험도 값을 정량화한다. 그러므로 각 개체의 위험도를 서로 비교할 수 있다는 장점이 있다. 그러나 기존 데이터들을 통한 경험적인 지식이 아니라 하나의 복잡도 메트릭 값에 의존한다는 단점이 있다. 위험도 메트릭 제작은 위험도에 가장 관련이 많은 기본 메트릭을 하나 선정하여 예측에 사용할 수도 있고 위험도와 관련이 있는 기본 메트릭들을 조합하여 하나의 조합 메트릭 형태를 사용할 수도 있다. 후자가 위험도에 관련된 여러 요인들을 고려할 수 있을 것 같지만 기본 메트릭들을 조합하는 것은 매우 주의를 요한다. 여러 메트릭들을 조합함으로써 각 구성 요소들의 특성을 잃어버릴 가능성이 있기 때문이다[12]. 메트릭을 만들어 위험도를 예측하는 연구로는 Zage의 연구[13], Agresti의 연구[14]와 데이터 바인딩 기법[15] 등이 있다.

두 가지 방향의 관련 연구들 모두 소프트웨어의 복잡도 메트릭이 프로그램의 오류의 분포와 관련이 있음을, 즉 복잡도가 높은 모듈일수록 오류가 발생할 가능성이 높다는 점을 가정하고 있다.

3. 제안 모델

본 논문에서 제안하는 새로운 위험도 예측 모델은 대표적인 무감독형 학습 신경망인 Kohonen SOM을 클러스터링 기법으로 사용하는 KSM이다.

3.1 KSM의 가정

KSM은 훈련데이터 집합을 사용하지 않으며 입력은 메트릭 벡터 형태이다. 만약 위험도를 하나의 메트릭으로 정량화 하여 하나의 값으로 나타낸다면 의미적인 문제가 없으나 메트릭 벡터 형태로 개체를 정량화 하면 벡터의 각 원소들은 여러 복잡도 요소들을 나타내지만 여러 개의 원소들로 구성된 벡터 값을 위험도라는 하나의 성질에 연관시키는 데는 어려움이 따른다. 그러므로 KSM과 같은 무감독형 학습 알고리즘을 사용하는 예측 모델은 몇 가지 가정이 필요하다.

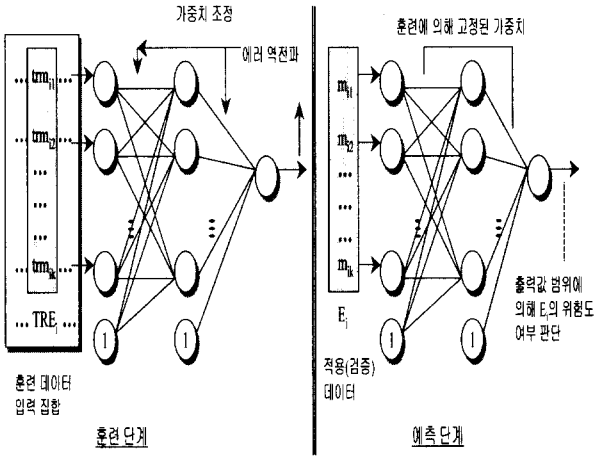
클러스터링이란 서로 유사한 개체들을 같은 클러스터에 넣고 상이한 개체들을 다른 클러스터에 넣음으로써 개체들을 여러 클러스터들로 나누는 것이다. SOM 알고리즘의 유사성의 기준은 유클리드 공간 거리이다. KSM은 메트릭 벡터 형태로 정량화된 설계 개체 집합을 몇 개의 유사한 클러스터로 나눈다. 여기서 생각해야할 두 가지는 나누어진 각 클러스터들이 비슷한 위험도를 나타내는가 하는 것과 실제 프로젝트의 설계 데이터들이 여러 개의 클러스터들로 반드시 나누어지는가 하는 것이다.

본 연구에서 각 클러스터들은 비슷한 위험도를 나타낼 가능성이 많다. 왜냐하면 정의된 SDL 메트릭들은 대부분 설계 개체의 특성을 모양이나 동작의 직접적인 수에 의해 나타내었기 때문이다. 그러면 각 클러스터들중 위험 클러스터와 비위험 클러스터는 어떻게 구분하는가? 이는 일반적인 분석뿐만 아니라 개발 집단의 성질이나 사용 기법 등 여러 다른 요소들에 의존하므로 자동화할 수 없으며 사람의 분석 과정이 필요하다. 설계 개체들의 데이터는 반드시 여러 개의 클러스터들로 나누어진다고 할 수는 없다. 따라서 KSM은 비슷한 위험도를 갖는 입력 벡터들은 유사하며, 입력 벡터들은 여러 개의 클러스터들을 형성한다는 가정을 둔다. 또한 KSM은 많은 입력 개체가 있는 경우에 의미가 있다. 너무 적은 개체들을 클러스터로 나누는 것은 의미가 없기 때문이다.

3.2 KSM 구조

기존 예측 모델 중 매우 좋은 성능을 보인다고 알려진 역전파 신경망 모델(BPM)을 사용한 위험도 예측 과정을 (그림 3)에 나타내었다. TRE_i 즉 (trm_{i1}, …, trm_{ik})는 훈련

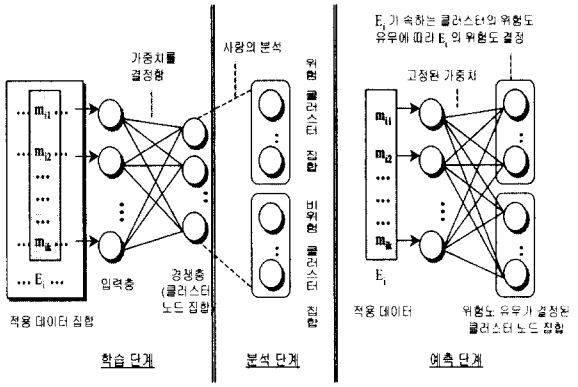
데이터 집합에 속한 개체의 메트릭 벡터를 나타내며 E_i 즉 $(m_{i1}, \cdots, m_{ik})$는 훈련 데이터를 통해 학습된 모델의 입력인 설계 개체의 메트릭 벡터를 나타낸다. 신경망 입력층의 각 뉴런은 메트릭 벡터를 구성하는 메트릭 하나를 의미하고 출력층은 위험 개체이냐 비위험 개체이냐를 결정하기 위해 하나의 뉴런으로 구성하였다. 또한 일반적인 역전파 신경망 구조를 따라 하나의 은닉층을 사용하였다. (그림 3)과 같이 BPM을 이용한 위험도 예측은 훈련 단계와 예측 단계의 두 단계로 나뉜다.



(그림 3) BPM 구조와 위험도 예측

SOM은 입력 자료들을 여러 클러스터로 그룹 짓는 클러스터링망의 하나이다. 클러스터링망의 출력 뉴런들은 각 클러스터를 나타내며 입력 뉴런들은 입력 벡터의 구성원들을 나타낸다. 각 출력 뉴런을 위한 가중치 벡터는 망이 각 클러스터를 위해 설정하는 것으로 입력 패턴들에 대한 표본 벡터의 역할을 한다. SOM은 역전파 신경망 모델과 달리 하나의 전방 경로를 사용하므로 구조상 수행이 상당히 빠르다는 장점을 가지고 있다. 그리고 연속적인 학습이 가능하여 입력 데이터의 통계적 분포가 시간에 따라 변하더라도 자동적으로 이러한 변화에 적응하게 된다. 즉 SOM은 자기 조직화를 통한 정확한 통계 모델이다. 본 논문은 신경망을 이용한 예측 모델과 그 사용 방법에 대한 것이기 때문에 신경망 알고리즘들에 대해서는 언급하지 않는다.

KSM을 이용한 위험도 예측은 BPM과는 달리 신경망 학습 단계, 분석 단계, 위험도 예측 단계의 세 단계로 나뉘며 이는 (그림 4)에 나타나 있다. 입력층의 각 뉴런은 BPM과 같이 설계 개체의 각 메트릭을 의미하며 출력층의 각 뉴런은 클러스터를 의미한다. 각 클러스터는 입력층과 클러스터 뉴런이 연결된 가중치 벡터로 표현된다.



(그림 4) KSM 구조와 위험도 예측

KSM은 훈련데이터 집합을 필요로 하지 않으며 학습 단계에서부터 현재 프로젝트 입력 데이터인 적용데이터집합을 사용한다. 출력층은 위험 그룹과 비위험 그룹을 나타내는 두개의 뉴런을 사용할 수도 있으나 위험 그룹에 여러 개의 패턴 형태가 존재할 수 있으므로 두 개보다 많은 클러스터 뉴런들을 두는 것이 바람직하다. SOM 알고리즘을 사용하여 학습이 끝나면 가중치가 결정된다. 분석 단계에서는 SDL 설계 및 메트릭 전문가들이 결과 클러스터들의 가중치값들을 분석하여 이들을 위험 클러스터 집합과 비위험 클러스터 집합으로 나눈다. 하나의 클러스터는 결정된 대표 가중치 값으로 표현되므로 벡터를 구성하는 메트릭 수, 즉 신경망의 입력 뉴런 수가 많을수록 분석에는 상당한 어려움과 기술이 요구된다.

분석 단계가 끝나면 학습된 신경망에 각 개체의 메트릭 벡터값을 넣어 그 개체가 속하는 클러스터의 위험도 여부에 따라 개체의 위험도 유무를 예측한다.

KSM의 문제점은 위험 개체들이 비슷한 복잡도 메트릭 벡터값들을 갖는다는 가정에 매우 의존적이라는 것과 분석 단계에 요구되는 어려움이다. 또한 SOM 알고리즘은 클러스터의 수를 학습 단계 이전에 미리 고정해야 한다. 클러스터 수가 너무 작으면 패턴 분류가 되지 않아 위험 그룹과 비위험 그룹들이 같은 클러스터에 속할 가능성이 있고, 클러스터 수가 너무 크면 사람의 분석 노력이 많이 들기 때문에 클러스터의 수를 적절히 결정하는 것은 학습 단계에서 매우 중요하다. 클러스터 수를 크게 하면 SOM은 빈 클러스터를 만들어 냄으로써 클러스터수가 많음을 암시하여 준다. 그러나 빈 클러스터가 많아지기 이전에 사람이 분석 노력 비용 등을 고려하여 적절한 클러스터수를 결정하여야 한다.

3.3 클러스터링 평가방법

KSM이 사용하는 SOM 알고리즘의 특성 중 또 다른 고

려사항은 적절한 클러스터 수를 결정하더라도 학습하는 경우마다 다른 학습 결과가 나온다는 것이다. 그러므로 여러 클러스터링 결과들 중 가장 좋은 결과를 학습 단계의 결과로 사용할 수 있도록 클러스터링 결과를 평가할 수 있는 기준이 필요하다. 클러스터링의 평가 기준들은 다음과 같다.

- 클러스터 내부의 응집도는 클수록 좋다.
- 클러스터간의 결합도는 작을수록 좋다.

이 두 기준들은 클러스터링 결과 형태에 대한 평가 기준이며 상호 연관성이 있다. 클러스터 수를 m, i번째 클러스터의 중심 벡터를 centeri, i번째 클러스터의 j번째 벡터를 xij라 하면 이들은 다음과 같은 측정치로 측정 가능하다.

$$\sum_{i=1}^{m}\sum_{j}|center_i - x_{ij}|^2 \quad (1)$$

클러스터의 중심 벡터는 학습이 끝난 후의 각 클러스터의 가중치 벡터를 사용한다. 식 (1)의 값이 작을수록 전체적으로 응집도는 높고 결합도는 낮은 좋은 클러스터링이 된다. 그러나 식 (1)은 같은 수의 클러스터를 갖는 여러 다른 클러스터링 결과들을 비교할 때는 적당하나 서로 다른 수의 클러스터를 갖는 클러스터링 결과들의 평가 비교에는 적당치 않다. 클러스터수가 커질수록 결과 클러스터는 응집될 가능성이 많지만 식 (1)의 바깥쪽 덧셈 항의 수는 많아진다.

4. 모의 실험

4.1 제작 데이터 집합

제안 모델의 유용성을 검증하기 위해 기존 분류 모델 중 우수하다고 알려진 BPM과 예측 정확도를 비교하는 실험을 행하였다. 설계 개체 유형으로는 SDL 시스템 분석 단계에서의 블록을 사용하였으며 하나의 블록을 정량화하는 메트릭 벡터는 (BRS, EBS, EBC, BP, BS, BR)[1]이다[4]. 선행연구[2]와 유사한 제약 조건을 가진 데이터 집합을 제작하였으며 각 블록의 위험도는 SDL을 이용한 통신 소프트웨어 시스템의 설계 경험이 있는 두명의 소프트웨어 공학자에 의해 결정되었다.

본 데이터 집합의 문제점은 제약 조건들에도 불구하고 데이터들을 난수로 발생시켰기 때문에 균등 분포의 성질을 갖는다는 것이다. 실제 프로젝트 데이터에서는 대부분의 블록들이 비위험 블록이고 위험 블록은 소수일 것이므로 두 그룹은 매우 상이한 분포를 따를 가능성이 많다.

1) 개체 정량화에 사용된 메트릭들에 대한 설명은 [4]를 참조하기 바란다.

훈련데이터 집합은 BPM의 학습에 필요하며 이는 BPM의 성능을 결정하는 가장 중요한 요인이 된다. 훈련데이터의 형태는 (BRS, EBS, EBC, BP, BS, BR, FP)로 적용 데이터에 FP가 첨가된 형태이다. FP는 모델의 출력값으로 1은 위험 개체를 0은 비위험 개체임을 나타낸다. 500개의 블록들의 데이터를 생성한 후 FP 값을 결정하여 이 중 300개를 선정해 훈련데이터 집합으로 하고 나머지 200개를 검증데이터 집합으로 하였다. 이를 훈련데이터 집합 1, 검증데이터 집합 1이라 하며 이를 총칭해 데이터 집합 1이라 한다.

데이터 집합 1은 일반적인 SDL 블록들의 성질을 가지고 있지만 FP를 결정하는 판단 기준의 경계 부분에 많은 유사한 블록들이 존재하였다. 그러므로 위험 블록과 비위험 블록을 나누는데 많은 판단의 어려움이 있었고 서로 유사한 입력 메트릭값 분포를 이루는 블록들이 근소한 차이에 의해 서로 다른 그룹에 속하게 결정되었다. 따라서 위험 그룹과 비위험 그룹간의 차이를 데이터 집합 1보다 크게 한 훈련데이터 집합 2와 검증데이터 집합 2를 제작하여 데이터 집합 2라 하였다. 이는 데이터 집합 1에서 판단이 애매했던 블록들의 입력 값들을 제약 조건을 만족하는 범위 내에서 고쳐 애매성을 없앤 것이다.

4.2 모델의 예측 정확도 비교

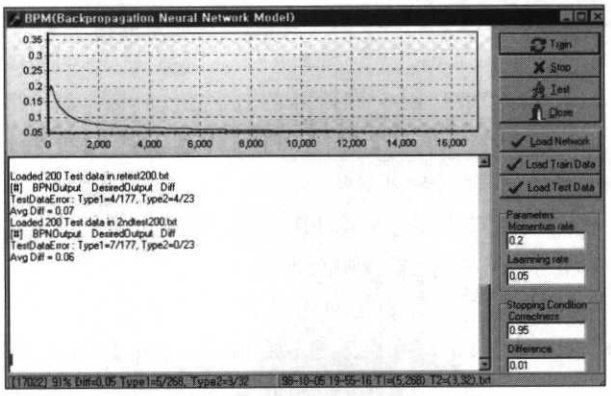
다음은 각 모델의 예측 정확도를 평가하기 위하여 본 실험에서 사용한 측정치들이다.

Type I 오류정보 : 비위험 개체를 위험 개체로 선정한 수 / 비위험 개체수

Type II 오류정보 : 위험 개체를 비위험 개체로 선정한 수 / 위험 개체수

Type I 오류는 비위험 개체를 위험 개체로 판단하는 경우고 Type II 오류는 위험 개체를 비위험 개체로 판단하는 경우이다. 전자는 소프트웨어 검증 단계의 비용을 높이지만 후자는 제품의 납품시기를 늦출 뿐만 아니라 생산 제품의 품질을 떨어지게 한다. 그러므로 후자가 전자보다 더욱 중요한 오류이다.

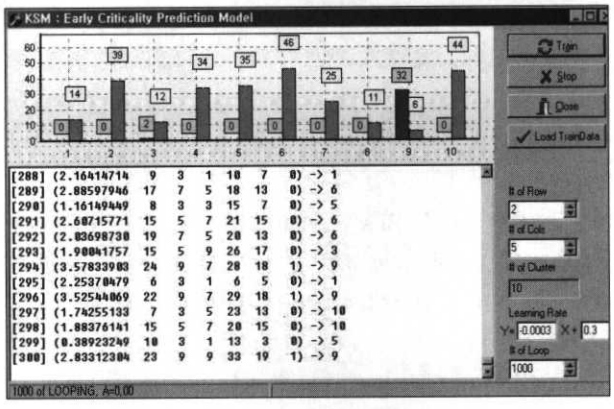
모델 훈련에 사용되는 여러 인자들은 훈련데이터 자체의 Type I, Type II 오류를 측정하여 좋은 성능을 보이는 인자들을 선택하였다. BPM은 학습률을 0.05, 운동량 변화율을 0.2로 하였으며 두 종류의 훈련데이터 집합을 사용한 모델에 두 종류의 검증 데이터 집합을 적용시켜 네 가지의 결과를 얻었다. (그림 5)는 훈련데이터 집합 1로 훈련시킨 BPM의 훈련 과정과 검증 데이터 집합 1의 예측 과정을 나타낸다.



(그림 5) BPM의 훈련과 위험도 예측

그래프의 x축은 입력 집합 전체의 훈련을 한 순환이라 할 때 순환값들을 나타낸다. 그래프 값은 각 순환에서 출력층 뉴런의 출력값과 해당 입력 벡터들의 실제 위험도값들의 차이들의 평균값을 나타낸 것이다. 그림 상의 훈련에서는 10000번 정도의 순환을 거친 후에는 0.05에 수렴하였으므로 훈련을 멈추었다.

(그림 6)은 훈련데이터 집합 2를 10개의 클러스터를 갖는 KSM에 적용한 결과들 중 하나이다. 실험 결과는 막대 그래프로 나타냈으며, 각 클러스터는 두 개의 막대로 구성된다. 각 막대에는 SDL 블록수가 나타나 있으며 왼쪽 막대가 실제 위험 블록수, 오른쪽 막대가 비위험 블록수를 나타낸다. 그림은 훈련데이터 집합 2의 34개의 위험 블록들이 클러스터 3과 클러스터 9에 2개와 32개로 나누어진 것을 나타낸다. 각 클러스터 가중치의 초기값은 난수로 초기화한 것보다 문제 영역 사이의 난수값으로 초기화한 모델이 약간 나은 결과를 보였다. 그러므로 KSM은 본 실험에서 사용하는 데이터 생성 알고리즘에 의해 생성된 값들을 가중치의 초기값으로 하였다. 또한 알고리즘의 순환수는 1000으로 하였으며, 학습률은 초기값을 0.3으로 하고 각 순환마다 0.0003씩 감소시켰다.



(그림 6) KSM의 훈련과 위험도 예측

KSM에서 위험 클러스터의 결정은 사람의 분석에 의해 수행되어야하지만 본 실험에서는 실험의 용이성을 위해 위험 블록이 속해 있으면서 비위험 블록수가 위험 블록수의 세배를 넘어가지 않는 클러스터를 위험 클러스터로 하였다. 결과적으로 이는 사람의 분석 결과와 유사한 결과가 되었다. (그림 6)의 클러스터 3은 위험 블록이 2개 있으나 비위험 블록이 12개 있으므로 위험 클러스터가 되지 않고 클러스터 9만 위험 클러스터가 된다. 결국 KSM의 위험도 예측 결과는 위험 클러스터수가 1개, 위험 블록수가 38개가 된다. 이 중 32개는 실제 위험 블록과 일치하지만 6개는 비위험 블록을 위험 블록으로 선정한 Type I 오류이다. 클러스터 3에 있는 위험 블록 2개는 위험 블록을 비위험 블록으로 선정한 Type II 오류가 된다.

실험 결과 생기는 위험 클러스터의 수와 분포 그리고 위험 블록수는 상당히 불규칙적인 형태를 보였다. 입력 데이터의 분포가 보다 확실히 구분된 훈련데이터 집합 2에 대해 KSM은 더 적은 위험 클러스터수와 정확한 위험 블록수 결과를 보였다. 이는 애매성이 적은 훈련데이터 집합 2의 위험 블록들이 더 적은 클러스터들에 밀집되며, 이 클러스터들에는 비위험 블록들이 많지 않음을 나타낸다.

〈표 1〉 BPM과 KSM의 예측 결과

모델 \ 판별오류		검증데이터 집합 1		검증데이터 집합 2	
		Type I	Type II	Type I	Type II
BPM	훈련데이터 집합 1	4/177	4/23	7/177	0/23
	훈련데이터 집합 2	6/177	11/23	0/177	0/23
KSM	10 클러스터	22.8/177	3.4/23	4.4/177	0.2/23
	15 클러스터	16.8/177	2.6/23	5.8/177	1.2/23
	20 클러스터	13.4/177	3.0/23	6.8/177	1.0/23
	25 클러스터	9.4/177	3.4/23	2.8/177	1.2/23

<표 1>은 BPM과 KSM의 예측 정확도 실험 결과를 축약한 것이다. KSM은 정해진 네 가지 클러스터 수에 따라 30번씩 실험을 하였으며 <표 1>에는 30개 결과들의 평균치를 나타내었다. 또한 각 결과들의 불규칙 정도를 보기 위해 평가 측정치인 식 (1)을 사용하여 하나의 클러스터링 결과를 선정하지는 않았다. BPM은 두 개의 훈련데이터 집합과 두개의 검증데이터집합에 대해 네 개의 결과를 나타내었지만 훈련데이터 집합 2로 훈련시킨 모델에 검증데이터 집합 1을 적용시키는 것은 의미가 없다. 왜냐하면 위험 그룹과 비위험 그룹의 차이가 명확한 데이터로 훈련된 모델이 차이가 매우 애매한 입력 집합에 대해 정확한 판단을 내리기 어렵기 때문이다. 표에서도 이 경우에 BPM은 매우

많은 오류를 냄을 볼 수 있다. 이러한 결과는 BPM같은 기존 모델 사용 시 훈련데이터 집합으로 사용하는 과거 프로젝트 데이터 집합의 분포가 현재 프로젝트의 적용 데이터 집합의 분포와 매우 유사하여야함을 의미한다. 훈련데이터 집합 2로 훈련시킨 모델에 검증데이터 집합 2를 적용한 경우 BPM은 오류를 하나도 내지 않았다.

실험 전의 예상으로는 클러스터수를 늘일수록 미세한 입력 패턴들이 작은 클러스터들을 형성하는 효과와 클러스터 내의 응집도가 높아지는 효과 때문에 보다 나은 결과를 내리라 예상했지만 실험 결과는 클러스터수를 증가시키는 것이 KSM의 예측 정확도에 그리 큰 영향은 미치지 않았음을 보여준다.

클러스터수를 증가시킬수록 결과 클러스터의 분석 노력이 많이 들며, 20개 정도의 클러스터부터는 SOM이 빈 클러스터를 만들어내기 시작했다. 알고리즘 순환수는 3000번으로 늘여서 실험하여 보았지만 비슷한 성능의 결과를 얻었다.

KSM은 Type I 오류는 BPM보다 상대적으로 컸으나 Type II 오류가 비슷하였으므로, 프로젝트에 미치는 위험이 적은 모델이라 할 수 있다. 즉 Type II 오류만 본다면 BPM과 비슷한 성능을 보였다.

5. 결 론

대형 소프트웨어 개발 프로젝트에서 초기 위험도 예측 모델은 효율적인 자원 할당과 재설계 부분의 자동 결정에 사용되므로 시스템 개발비용을 낮추는 데 큰 몫을 하고 있다. 하지만 판별 분석, 분류 트리, 역전파 신경망 등을 이용한 기존 모델 대부분은 과거 프로젝트에서 얻어진 훈련데이터집합을 필요로 하는 매우 큰 문제점이 있었다. 또한 설사 훈련데이터 집합이 존재한다 하더라도 과거 프로젝트 데이터와 현재 프로젝트 데이터가 매우 유사하여야한다는 제약점을 가지고 있다. 이러한 문제점들은 위험도 예측 모델을 꼭 필요로 하는 대규모 개발집단에서 조차 위험도 예측 모델의 사용을 어렵게 만들고 있다. 이러한 문제점들을 해결하기 위해 본 논문에서는 Kohonen SOM 신경망을 이용하여 훈련데이터집합을 사용하지 않는 위험도 예측 모델인 KSM을 제안하였다. KSM의 성능 평가를 위해 KSM과 기존 모델들 중 예측 정확도 면에서 매우 우수하다는 역전파 신경망 모델 즉 BPM을 구현하여 예측 정확도를 비교하였으며, 두 모델은 Type I 오류 측면에서는 BPM이 우수하였으나 Type II 오류 측면에서는 KSM이 BPM에 근접한 성능을 보였다. 그러므로 현재 프로젝트와 유사한 과거 프로젝트의 훈련데이터집합이 있는 집단에서는 BPM의 사용이 바람직하나 그렇지 않은 대부분의 개발 집단에서는 KSM이 대체 모델로 사용될 수 있다.

향후 과제로는 KSM의 클러스터 수를 보다 자연스럽게 결정할 수 있는 방법의 개발과 클러스터 수를 고정하지 않는 클러스터링 알고리즘을 모델에 사용해 보는 것이다.

참 고 문 헌

[1] J. Tian, A. Nguyen, C. Allen and R. Appan, "Experience with identifying and characterizing problem-prone modules in telecommunication software systems," J. Systems Software, Vol.57, pp.207-215, 2001.

[2] 홍의석, 권용길, "퍼지 분류를 이용한 초기 위험도 예측 모델", 정보처리학회논문지, 제7권 제5호, pp.1401-1408, 2000.

[3] N. Ohlsson and H. Alberg, "Prediction Fault-Prone Software Modules in Telephone Switches," IEEE Trans. Software Eng., Vol.22, No.12, pp.886-894, Dec., 1996.

[4] 홍의석, 홍성백, 김갑수, 우치수, "SDL 설계 복잡도 메트릭 집합", 정보과학회논문지(B), 제24권 제10호, pp.1053-1062, 1997.

[5] 홍의석, 정명희, "SDL 메트릭 집합의 분석적 검증", 정보처리학회논문지, 제7권 제4호, pp.1112-1121, 2000.

[6] T. M. Khosgoftaar and E. B. Allen, "Early Quality Predictin : A Case Study in Telecommunications," IEEE Software, Vol.13, No.1, pp.65-71, Jan., 1996.

[7] T. M. Khoshgoftaar and D. L. Lanning, "A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase," J. Systems Software, Vol.29, pp.85-91, 1995.

[8] L. C. Briand, J. Daly, V. Porter and J. Wüst, "Predicting fault-prone classes with design measures in object-oriented systems," Proc. Int'l Symp. Software Reliability Engineering, pp.334-343, 1998.

[9] L. C. Briand, V. R. Basili and C. J. Hetmanski, "Developing interpretable models with optimized set reduction for identifying high-risk software components," IEEE Trans. Software Eng., Vol.19, No.11, pp.1028-1044, 1993.

[10] K. E. Emam, S. Benlarbi, N. Goel and S. N. Rai, "Comparing case-based reasoning for predicing high risk software components," J. Systems Software, Vol.55, pp.301-320, 2001.

[11] Y. Liu and T. M. Khoshgoftaar, "Genetic programming model for software quality classification," Proc. IEEE Int'l Symp, High Assurance Systems Engineering, pp.127-136, 2001.

[12] N. Fenton, "Software Measurement : A Necessary Sci-

entific Basis," IEEE Trans. Software Eng., Vol.20, No.3, pp.199-206, March, 1994.

[13] W. M. Zage and D. M. Zage, "Evaluating Design Metrics on Large-Scale Software," IEEE Software, pp.75-80, July, 1993.

[14] W. W. Agresti and W. M. Evanco, "Projecting Software Defects From Analyzing Ada Designs," IEEE Trans. Software Eng., Vol.18, No.11, Nov., 1992.

[15] R. W. Selby and V. R. Basili, "Analyzing error-prone system structure," IEEE Trans. Software Eng., Vol.17, No.2, pp.141-152, Feb., 1991.

홍 의 석

e-mail : hes@cs.sungshin.ac.kr

1992년 서울대학교 계산통계학과(학사)

1994년 서울대학교 대학원 계산통계학(이학석사)

1999년 서울대학교 대학원 전산과학과(이학박사)

1999년~2001년 안양대학교 디지털미디어학부 교수

2002년~현재 성신여자대학교 컴퓨터정보학부 교수

관심분야 : 메트릭 기반 소프트웨어 품질 예측 모델, 웹 기반 컴포넌트 응용 기술 등