Support Vector Machine을 이용한 초기 소프트웨어 품질 예측*

홍 의 석**

Early Software Quality Prediction Using Support Vector Machine*

Euyseok Hong**

■ Abstract ■

Early criticality prediction models that determine whether a design entity is fault-prone or not are becoming more and more important as software development projects are getting larger. Effective predictions can reduce the system development cost and improve software quality by identifying trouble-spots at early phases and proper allocation of effort and resources. Many prediction models have been proposed using statistical and machine learning methods.

This paper builds a prediction model using Support Vector Machine(SVM) which is one of the most popular modern classification methods and compares its prediction performance with a well-known prediction model, BackPropagation neural network Model(BPM). SVM is known to generalize well even in high dimensional spaces under small training data conditions. In prediction performance evaluation experiments, dimensionality reduction techniques for data set are not used because the dimension of input data is too small. Experimental results show that the prediction performance of SVM model is slightly better than that of BPM and polynomial kernel function achieves better performance than other SVM kernel functions.

Keyword : Software Quality, Prediction Model, Support Vector Machine

논문투고일 : 2011년 03월 19일 논문수정완료일 : 2011년 05월 16일 논문게재확정일 : 2011년 06월 11일

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2010-0021902).

** 성신여자대학교 IT학부 교수

1. 서 론

소프트웨어 운영 환경이 다양해지고 새로운 개발 방법들과 개발 도구들이 등장하고 있지만 많은 시간과 개발 인력이 필요한 대형 소프트웨어 개발의 성패는 주로 소프트웨어 프로세스 관리에 달려 있다. 소프트웨어 프로세스 관리는 한정된 예산과 자원을 이용하여 가장 고품질의 소프트웨어를 제작 및 유지보수 하는 것을 목표로 하며 이를 위하여 적절한 자원의 배치를 도와주는 소프트웨어 품질 예측 모델들이 사용된다. 예측 모델들은 주로 분석이나 설계와 같은 개발 초기 단계의 산물을 사용하여 소프트웨어 개발 산물의 품질 인자들을 예측하는 모델들이다. 주로 사용되는 모델의 입력들은 설계 명세를 정량화한 설계 메트릭 집합이며 예측 품질 인자들은 위험도 또는 유지보수성이었다. 설계 개체의 위험도(criticality)란 개체가 구현되었을 때의 결함경향성을 의미하며[1] 품질 예측 모델에 대한 연구들은 대부분 위험도에 대한 것들이었다. 그 이유는 구현 소프트웨어에서 결함을 많이 일으킬 부분들을 초기 단계에서 미리 찾아 적절한 자원 할당을 가능케 함으로써 소프트웨어 개발 전체 비용을 낮추고 구현된 소프트웨어의 품질을 매우 높일 수 있기 때문이다. 따라서 본 논문의 소프트웨어 품질 예측 모델은 설계 명세를 입력으로 한 위험도 예측 모델을 의미한다.

위험도 예측에 관한 연구는 결함수를 예측하는 연구보다 개체의 결함경향성 여부를 판단하는 것이 주류를 이룬다. 설계 개체를 보고 정확한 결함수를 예측하는 것이 어려운 것도 있지만 위험도가 높은 문제 지점들을 찾는 것만으로 프로세스 관리에서 필요한 정보로 충분하기 때문이다. 따라서 많은 연구들에서 제안한 위험도 예측 모델들은 대부분 메트릭 벡터들로 설계 개체들을 정량화 한 후 이들을 위험 그룹과 비위험 그룹으로 분류하는 분류 모델들이었다. 이들은 주로 과거에 얻은 위험도 결과 데이터를 학습하여 현재 프로젝트에 사용하는 훈련 모델들이었으며 훈련 알고리즘으로는 복잡한 통계 기법들이나 인공지능 기법들을 사용하였다.

본 논문의 목적은 최근 비선형 관계들을 모델링하는 응용 분야에 많이 사용되는 커널기반 분류 알고리즘인 Support Vector Machine(SVM)[2]을 이용한 모델을 제작하고 튜닝하여 많은 연구들에서 좋은 결과를 보인다고 알려진 오류역전파 신경망을 사용한 모델과 예측 정확도 성능을 비교해보는 것이다. SVM은 작은 크기의 훈련 집합으로도 고차원의 문제 공간을 일반화하는 능력이 탁월하다고 알려져 있으므로[3] 정성적으로도 품질 예측에 적합한 분류 알고리즘이다.

제 2장에서는 네가지 형태로 분류된 기존의 위험도 예측 모델들과 SVM을 이용한 연구들을 살펴보고 제 3장에서는 SVM 알고리즘과 제안 모델 및 비교 모델 구축에 사용된 파라미터 초기화를 설명한다. 제 4장에서는 성능 실험에 사용할 데이터 집합과 훈련 및 검증 결과로 구성된 예측성능 실험 결과에 대해 언급하고 제 5장에는 결론에 대해 기술한다.

2. 초기 소프트웨어 품질 예측

2.1 예측 모델의 분류

최근까지 수많은 위험도 예측 모델들에 대한 연구가 발표되었지만 그들을 체계적으로 분류한 연구는 거의 없다. 2009년 [4]는 그 때까지 발표된 결함경향성 예측 모델 논문들을 참조하여 각 연구들이 사용한 메트릭들, 방법(모델)들, 데이터 집합들을 기준으로 하여 각 연구들의 특성들을 분석하였다. 하지만 이는 예측 모델을 분류한 연구는 아니었고, 단순히 세가지 기준에 어떤 것들이 많이 사용되었는지 등을 살펴보는 연구에 불과하였다.

하지만 그 후 [5]는 위험도 예측 모델을 분류하기 위한 분류 프레임워크를 정의하여 다음과 같은 4가지 형태의 의미 있는 모델 타입을 정의하였다.

- *VI-SL*(Vector Input-Supervised Learning)

메트릭 벡터 입력 모델이면서 감독형 모델인 경우

- *VI-UL*(Vector Input-Unsupervised Learning) 메트릭 벡터 입력 모델이면서 비감독형 모델인 경우
- *SI-AC*(Scalar Input-Automatic Classification) 스칼라 메트릭 입력 모델이면서 비감독형 모델인 경우
- *SI-HC*(Scalar Input-Human Classification) 스칼라 메트릭 입력 모델이면서 사람이 분류하는 모델인 경우

4개의 타입은 모델 입력 메트릭 형태와 과거 프로젝트 데이터의 필요 유무라는 두 개의 기준에 의해 정의되었다. 스칼라 메트릭 입력이란 하나의 기본 메트릭이나 조합 메트릭 값으로 위험도를 추정하는 모델을 의미하며 메트릭 벡터 입력이란 여러 기본 메트릭으로 이루어진 벡터를 입력으로 하는 모델을 의미한다. 감독형 모델은 모델을 학습시킬 훈련 데이터가 필요한 모델이고 비감독형 모델은 훈련 데이터가 필요 없는 모델이다.

품질 예측 모델에 대한 연구 대부분은 VI-SL 모델에 대한 것들이었으며, 본 논문의 제안 모델인 SVM 이용 모델 역시 VI-SL 모델이다. 기존에 사용된 훈련 알고리즘들은 판별분석, 로지스틱 회귀분석[6] 등과 같은 통계 기법이나 분류트리[7], 역전파 신경망[8] 등과 같은 인공지능 기법들이다. VI-SL 모델은 반드시 훈련 데이터 집합을 필요로 하지만 대부분의 개발 집단은 이를 보유하고 있지 않다는 치명적인 문제점을 가지고 있다. 또한 설사 보유하고 있더라도 훈련 데이터가 현재 프로젝트에서 얻은 것이 아니라면 훈련 데이터를 얻은 과거 프로젝트와 예측 모델을 사용하려는 현재 프로젝트가 매우 유사하여야한다.

이 같은 문제점을 해결하기 위해 제안된 모델이 VI-UL 모델이다. 훈련 지식 없이 현재의 벡터 데이터만으로 개체의 위험도를 알아내는 것은 매우 어렵기 때문에 VI-UL 모델에 대한 연구는 거의 없다. [9]은 SOM 신경망을 사용하여 입력 벡터들의 분포를 보고 서로 유사한 입력 부분들을 묶어 클러스터들을 나눈 다음, 나누어진 클러스터들을 전문가가 보고 각 클러스터들의 성질(위험도 여부)을 결정하는 모델을 제안하였다. [10]은 K-means와 Neural-Gas 클러스터링을 사용하여 이와 유사한 모델을 제안하였으며 후속 연구로 [11]은 더 정확한 K-means 클러스터링을 위해 초기 클러스터 위치를 지정해주는 기법을 사용하였다.

초기의 위험도 예측 연구들에는 LOC, McCabe의 메트릭, Halstead의 메트릭, C&K 메트릭 등을 이용하여 하나의 메트릭 값으로 위험도를 예측하려는 노력들이 있었다. 이와 같은 SI 모델은 위험도를 결정하는 측정치로 하나의 기본 메트릭을 사용할 수도 있고 기본 메트릭들을 조합한 조합 메트릭 형태를 사용할 수도 있으나, 전자는 한가지 측면만을 고려한다는 것과 후자는 여러 기본 메트릭을 조합함으로써 이를 구성하는 기본 메트릭들의 성질을 잃을 수 있다는 문제점이 있다.

2.2 SVM 기반 모델들

SVM에 대한 관심이 높아지면서 [12]는 처음으로 결함경향성 유무 예측에 SVM을 사용하였다. 직접 수집한 데이터 모듈들을 가지고 각 모듈에 가해진 수정 수에 따라 결함경향성을 결정하였으며, 203개 모듈 중 89개의 결함경향 모듈을 결정하였다. 메트릭은 LOC, McCabe, Halstead 메트릭들을 사용하였으며 차원 축소를 위해 PCA(Principal Component Analysis)를 사용하였다. 비교 모델은 QDA(Quadratic Discriminant Analysis), CART(Classification and Regression Tree)였으며, 예측 성능을 높이기 위해 결함경향성이 결정되지 않은 데이터를 훈련 데이터와 함께 사용하는 TSVM(Transitive SVM)도 사용되었다. 실험 결과 예측 성능은 TSVM, SVM, QDA, CART순으로 좋았으며 PCA 사용 유무는 SVM 성능에 영향을 못 미쳤다. 이 연구의 중요한 시도는 위험 제어를 도입

한 것으로 중요한 예측 오류인 Type II 오류를 줄이기 위해 슬랙 변수에 대한 페널티 변수 C를 데이터 구간별로 두 개로 하여 실험했다는 것이다. 이 연구의 문제점은 사용 데이터 집합의 결함경향 모듈의 크기가 44%로 10% 남짓한 일반적인 데이터 집합의 결함경향성 모듈의 크기보다 너무 크다는 것, 좋은 성능을 보인 TSVM은 훈련 시간 오버헤드가 매우 크다는 것, 페널티 변수를 나누어 시도한 위험 제어 방법은 전체적인 예측 성능과 Type I 오류를 급격히 증가시킨다는 것이다.

[13]은 PCA의 문제점인 차원 축소 결과를 이해하기 어렵다는 것을 해결하기 위해 SA(Sensitivity Analysis)를 사용하였으며 신경망과 SVM으로 구현한 모델의 성능을 비교하였다. SA는 훈련 데이터를 신경망으로 학습시킨 후 학습 모델의 출력값을 이용해 각 메트릭에 대한 SCI를 구하여 SCI가 낮은 메트릭들을 삭제시키는 방법이며, 이를 통해 축소된 데이터 집합을 가지고 다시 신경망과 SVM에 적용하여 예측 모델을 만든다. 데이터는 NASA IV&V JM1 프로젝트를 사용하였으며 예측 비교 결과는 SVM 이용 모델의 예측 적중률이 87.4%로 신경망의 72.61%보다 훨씬 높았다. 하지만 예측 적중률 외에는 예측 오류 등에 관련된 여러 측정치들을 실험에 제시하지 않음으로 두 모델의 정확한 비교가 어렵다는 문제점이 있다.

[3]은 NASA IV&V CM1, PC1, KC1, KC3 프로젝트 데이터를 이용하여 SVM 모델과 8개의 다른 모델들을 비교하였다. 차원 축소는 CFS(Correlation-based Feature Selection)를 사용하였고 결과 분석을 위해 예측 성능 평가에 많이 사용되는 측정치들인 Accuracy, Precision, Recall, F-measure를 사용하였다. 실험 결과 SVM 모델은 대체로 Recall은 다른 모델들보다 낮고, Precision은 좋지 않았으며 다른 두 측정치는 특별히 의미 있는 결과를 보이진 않았다. 모델 각각을 보면 SVM과 유의미하게 비슷한 결과를 낸 모델은 역전파신경망 모델이었고, BBN(Bayesian Belief Network)과 NB(Naive Bayes)는 Precision 측정치로는 모든 데이터 집합에서 SVM보다 나은 결과를 보였다. Precision은 예측한 위험 모듈 수에 대한 맞춘 위험 모듈 수의 비율이고, Recall은 실제 위험 모듈 수에 대한 맞춘 위험 모듈 수의 비율이다. 하지만 이는 품질 관리에 중요한 영향을 미치는 Type II 오류 정보를 나타내지 못한다는 문제점이 있다. Type II 오류는 실제 위험 모듈 수에 대한 잘못 맞춘 위험 모듈 수의 비율이다. 이것을 넣어 실험했더라면 좀 더 나은 분석이 가능했을 것이다.

[14]는 다중 클래스 분류를 지원하는 SVM을 이용하여 모듈의 잠재위험수준을 예측하는 모델을 제안하였다. 데이터는 NASA IV&V CM1, PC1 프로젝트 데이터를 사용하였으며, 실험 환경은 MATLAB과 SVMmulticlass를 이용하고, 차원 축소는 [3]과 [13]의 결과를 이용하였다. 모델의 출력인 위험수준은 요구사항의 위험수준정보와 구현 모듈의 오류수를 이용하여 1~5의 값을 갖는 식으로 정의하였으며, 4이상의 출력이면 위험수준 모듈로 보았다. 이 연구의 문제점은 [13]은 JM1 프로젝트를 이용한 연구인데 이와 다른 데이터 집합을 사용하면서 해당 차원 축소 집합을 이용한 것은 부적절하다는 점과 다른 예측 모델과의 비교 연구가 없다는 점, SVM에서 거의 사용하지 않는 Linear 커널을 사용한 점, 전체적으로 타 실험에 비해 분류오류율이 높게 형성되었다는 것이다.

3. 제안 모델

3.1 기계 학습에 기반한 분류 모델

d차원 입력 메트릭 벡터 x_i와 출력값 y_i의 쌍인 (x_i, y_i)가 n개 있는 훈련 데이터 집합은 다음과 같이 표현된다. 분류 모델을 위한 훈련 집합이므로 출력은 1 또는 -1 두 가지로 된다.

$$(\boldsymbol{x}_i, y_i), i = 1, \cdots, n \quad (1)$$
$$where\ \boldsymbol{x}_i \in R^d,\ y_i \in \{\pm 1\}$$

기계 학습에 기반하여 분류 문제를 푸는 것은

훈련 데이터 집합을 가장 잘 학습하여 다음과 같은 최적화된 모델 f를 구하는 것이다.

$$f: R^d \mapsto \{\pm 1\}(\boldsymbol{x}_i, y_i) \in R^d \times \{\pm 1\} \quad (2)$$
$$(\boldsymbol{x}_i, y_i) \in R^d \times \{\pm 1\} \ \ i = 1, \cdots, n$$

최적화된 f를 찾기 위해 주어진 훈련 데이터 집합에 대한 f의 학습 오류를 줄이는 것이 중요하다. 하지만 주어진 훈련 데이터에 대해서만 학습 오류를 줄이는 것은 f를 더욱 복잡하게 만들고 이는 학습 데이터 이외의 다른 일반 데이터에 f를 적용하였을 때의 오류를 증가시킨다. 이 문제를 오버피팅이라 하며, 훈련 데이터 집합 외의 일반 데이터에서도 좋은 성능을 나타내는 것을 일반화라 한다. 교차검증(cross validation)은 오버피팅을 줄이기 위한 대표적인 방법이며 훈련 시 훈련 데이터 집합 외에 검증 데이터 집합을 함께 사용하여 일반화된 f를 찾는다. [13]은 일반적인 이중 교차 검증을 사용하였고, [3]은 10개로 나눈 다중 교차 검증을 사용하였다.

3.2 Support Vector Machine

SVM은 일반화 오류를 최소화하는 SRM 원리를 사용하는 커널기반 학습 알고리즘이다. 비교적 적은 학습 데이터를 가지고도 고차원 공간에서 일반화되며, 이는 SVM이 다른 모델들과 달리 차원 문제에 영향을 덜 받는다는 것을 의미한다.

3.2.1 선형 SVM

선형 SVM의 경우 훈련 데이터 집합은 선형으로 분리 가능한 경우와 분리가능하지 않은 경우로 나뉜다. 분리 가능한 경우는 분류 오류가 없다는 의미이며, 분리하는 의사결정경계에서 가장 가까운 두 데이터를 지나는 평행한 초평면 사이의 거리(마진)를 가장 크게 하는 최대 마진 분류기를 찾는 것이 목표이다. SRM 원리에 의해 마진이 클수록 최적화되고 일반화 오류가 적은 의사 결정 경계이기 때문이다. 만약 훈련 데이터 집합이 두 집합 A, B로 선형분리 가능하면 다음을 만족하는 (w, b)가 존재한다.

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq +1 \ \text{ for all } \boldsymbol{x}_i \in A, \ y_i = 1$$
$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 \ \text{ for all } \boldsymbol{x}_i \in B, \ y_i = -1$$

이는 다음과 같이 정리가능하다.

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \ \ i = 1, \cdots, n \quad (3)$$

$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$과 $\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$로 두 평행한 초평면을 표현할 수 있고 이들 사이의 마진은 $\frac{2}{\| w \|}$이므로 결국 선형 SVM의 학습은 (3) 조건하에서 다음 최적화 문제를 푸는 것이다.

$$\underset{w, b}{minimize} \frac{1}{2} \| \boldsymbol{w} \|^2 \quad (4)$$

이 문제는 표준 라그랑즈 이중 문제 풀이기법을 사용하여 다음과 같은 이중 제곱 프로그래밍 문제로 변환된다. 여기서 λ_i는 라그랑즈 승수이며 지지도 벡터 상에서는 0이고 아닌 경우에는 0보다 크게 된다.

$$maximize \left\{ \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} \right\} \quad (5)$$

$$s.t. \ \sum_{i=1}^{n} \lambda_i y_i = 0, \quad \lambda_i \geq 0 \ \ i = 1, \cdots, n \quad (6)$$

이 최적화 문제를 풀면 다음과 같은 의사결정 모델을 얻을 수 있다.

$$f(\boldsymbol{x}) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x_i} \cdot \boldsymbol{x} + b \right) \quad (7)$$

오류 없이 선형으로 분리 가능하지 않은 경우는 식 (3)의 제한 조건들을 만족하지 않는다. 그러므로 최적화 문제는 다음과 같이 양의 값을 갖는 슬

랙변수 ξ_i을 추가하여 오류 데이터들을 수용할 수 있도록 완화되어야한다.

$$\underset{w,\,b,\,\xi}{minimize}\left\{\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\xi_i\right\} \quad (8)$$

$$s.t.\quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)\geq 1-\xi_i \quad i=1,\cdots,n \quad (9)$$

C는 마진의 최대화와 분류 오류의 최소화의 트레이드오프를 조정하는 비용 매개변수이다. 이 최적화 문제의 풀이는 식 (6)의 라그랑즈 승수의 범위가 다음과 같이 변화하는 것을 제외하곤 분리 가능한 경우와 같다.

$$\sum_{i=1}^{n}\lambda_i y_i=0,\quad 0\leq\lambda_i\leq C\quad i=1,\cdots,n \quad (10)$$

3.2.2 비선형 SVM

비선형 의사결정경계를 갖는 데이터를 SVM으로 분류하는 일반적인 방법은 비선형 매핑 Φ를 이용해 입력 데이터를 선형 의사결정경계를 사용할 수 있는 보다 고차원 공간으로 변환시키는 것이다. 만약 적절한 를 찾았다면 비선형 SVM은 식 (4)를 다음 조건 하에서 푸는 것이 된다.

$$y_i(\boldsymbol{w}\cdot\boldsymbol{\Phi}(\boldsymbol{x}_i)+b)\geq 1\quad i=1,\cdots,n \quad (11)$$

이를 선형 SVM과 같은 방식으로 풀면 다음과 같은 모델을 얻을 수 있다.

$$f(\boldsymbol{x})=sign\left(\sum_{i=1}^{n}\lambda_i y_i\boldsymbol{\Phi}(\boldsymbol{x}_i)\cdot\boldsymbol{\Phi}(\boldsymbol{x})+b\right) \quad (12)$$

하지만 이러한 변환 기법은 선형 분리를 가능하게 하는 Φ를 찾기 어렵고, 고차원 공간에서 최적화 문제를 풀기 때문에 많은 계산 부하가 거릴 수 있다는 문제점이 있다. 이 문제점을 해결하기 위해 변환된 공간에서 두 입력 사이의 유사성을 계산하는 커널 트릭 기법이 사용되었고 이를 계산하는 함수가 다음과 같은 커널 함수 K이다.

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{\Phi}(\boldsymbol{x}_i)\cdot\boldsymbol{\Phi}(\boldsymbol{x}_j) \quad (13)$$

커널 함수를 사용한 최종 의사결정모델은 다음과 같다.

$$f(\boldsymbol{x})=sign\left(\sum_{i=1}^{n}\lambda_i y_i K(\boldsymbol{x}_i,\boldsymbol{x})+b\right) \quad (14)$$

다음은 가장 많이 쓰이는 네 가지 커널함수들이다.

Linear : $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i\cdot\boldsymbol{x}_j$

Polynomial : $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=(1+\boldsymbol{x}_i\cdot\boldsymbol{x}_j)^{degree}$

RBF : $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2),\ \gamma>0.$

Sigmoid : $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(a\boldsymbol{x}_i\cdot\boldsymbol{x}_j+b)$

Linear는 선형분류기이므로 비선형적인 데이터 집합에는 사용하기 어렵다. 커널함수 중 가장 널리 사용되는 RBF(Radial Basis Function)는 수식에서 γ를 $1/2\sigma^2$로 한 가우시안 형태로 많이 사용된다. Polynomial은 *degree*가 높아질수록 분류 경계면이 복잡해지고 오버피팅 될 가능성도 높아진다. Sigmoid는 인공신경망의 전이 함수로 많이 사용된다.

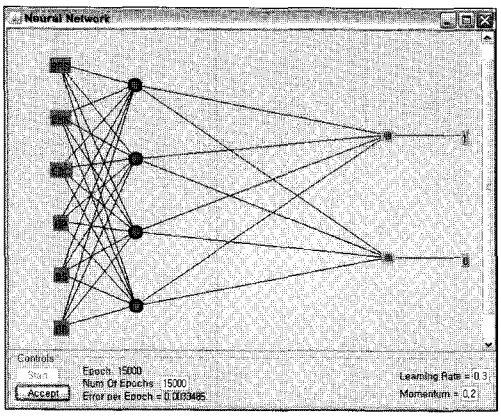
3.3 파라미터 초기화

본 논문에서는 SVM 모델과 오류역전파 신경망 모델인 BPM의 예측 성능을 비교한다. SVM 모델은 WEKA[1]의 구현을 사용하였다. BPM은 선행연구 [9]에서 구현한 모델은 BPM_old로 WEKA에서 제공하는 BPM을 튜닝한 모델을 BPM_new로 표기한다.

SVM의 파라미터는 비용 매개변수 C는 1.0, RBF

1) WEKA(Waikato Environment for Knowledge Analysis), http://www.cs.waikato.ac.nz/~ml/weka/.

함수의 *gamma*는 1/max_index인 0.167, Polynomial 함수의 *degree*는 3으로 하였다. BPM_old의 구조는 하나의 은닉층을 가지는 세 개의 층으로 구성되며 출력층은 하나의 노드로 구성하여 출력값의 범위에 따라 입력 개체의 결함경향성을 결정하게 하였다. BPM_old의 학습률은 0.05, 운동량 변화율은 0.2로 하고 훈련 데이터에 대해 10,000번 정도의 순환을 통해 학습하였다.



[그림 1] BPM_new 훈련 결과

BPM_new 구조는 은닉층은 하나이지만 출력층의 노드를 결함경향성이 1인 노드와 0인 노드 두 개로 하였다. 학습률은 0.3, 운동량 변화율은 0.2로 하였으며 13,000순환 후에는 오류값이 수렴하였으므로 15,000순환만큼 훈련시켰다. 은닉층 노드의 수는 WEKA에서 제공하는 기본 와일드카드 중 일반적인(입력층 노드수+출력층 노드 수)/2 공식을 사용하여 4로 하였으며 모든 노드는 일반적인 시그모이드 함수를 사용하였다. [그림 1]은 BPM_new의 위상을 나타내며 출력층의 노드가 두 개인 것을 확인할 수 있다.

4. 실험

4.1 데이터집합

실험에 사용한 데이터 집합은 [9]에서 BPM_old에 사용한 데이터 집합을 사용하였다. 모델의 입력 대상은 객체지향 실시간 시스템 명세 언어인 SDL로 작성한 설계 명세이며 설계 블록을 정량화하는 메트릭 벡터는(BRS, EBS, EBC, BP, BS, BR)이다. 이는 SVM 모델과 BPM의 입력 벡터 형태가 되며 각 메트릭 벡터 값에 대해 출력값 FP가 존재한다. FP는 위험 개체는 1, 비위험 개체는 0값을 갖는다.

훈련 1은 300개의 개체로 구성된 훈련 데이터 집합이고 검증 1은 200개의 개체로 구성된 훈련 모델의 성능을 검증하기 위한 데이터 집합이다. 훈련 2, 검증 2는 위험 그룹과 비위험 그룹간의 차이를 크게 해 위험도 판단이 애매한 개체들을 없앤 데이터 집합을 나타낸다.

입력 데이터 집합에 대해 PCA나 SA같은 차원 축소 작업은 수행하지 않았다. NASA 데이터 집합을 사용한 SVM 모델들은 입력 벡터의 크기가 21이었으므로 차원의 크기 문제가 발생할 수 있었지만 본 실험의 입력 벡터의 크기는 6으로 축소할 이유가 없다. 또한 SVM은 작은 훈련 데이터 집합으로도 고차원 공간에서 일반화가 잘 된다고 알려져 있으므로 SVM의 학습 능력은 입력 공간의 차원에 어느 정도 독립적일 수 있다. 실제로 [12]의 실험 결과에서 SVM 사용 모델은 PCA를 사용한 SVM 모델과 결과면에서 유의미한 차이가 없었으며 Type II 오류는 오히려 전자가 적었다.

4.2 훈련 결과

예측 모델의 예측 오류는 두 가지 형태가 존재한다. Type I 오류란 비위험 개체를 위험 개체로 판단하는 오류이고, Type II 오류는 위험 개체를 비위험 개체로 판단하는 오류이다. 전자보다는 후자가 제품의 품질을 떨어지게 하는 중요한 원인이 되므로 후자가 더욱 중요한 오류이다. 다음은 두 오류를 나타내는 측정치들이다.

- Type I 오류 : 비위험 개체를 위험 개체로 선정한 수/비위험 개체수

• Type II 오류 : 위험 개체를 비위험 개체로 선정한 수/위험 개체수

<표 1>은 훈련 1과 훈련 2에 대한 훈련 결과를 나타낸다. 즉, 훈련이 완료된 상태에서 훈련 데이터 집합에 대한 훈련 오류 정보를 나타낸다. BPM은 더 나은 결과를 보인 정규화 데이터를 사용한 경우를 나타냈으며 SVM은 정규화와 비정규화 데이터를 사용한 경우들을 각각 나타내었다. SVM의 커널 함수는 일반적으로 많이 사용되는 RBF와 Polynomial 함수를 사용한 결과를 나타내었다.

〈표 1〉 훈련 결과

모델 \ 훈련 데이터		훈련 1		훈련 2	
		Type I	Type II	Type I	Type II
SVM (NotNormalize)	RBF	1/268	6/32	0/266	4/34
	Polynomial	0/268	0/32	0/266	0/34
SVM (Normalize)	RBF	1/268	17/32	2/266	0/34
	Polynomial	0/268	16/32	2/266	1/34
BPM_old		5/268	3/32	2/266	0/34
BPM_new		1/268	0/32	1/266	0/34

결과는 비정규화 데이터에 Polynomial 함수를 사용한 SVM이 훈련 1과 훈련 2에 대해 단 하나의 훈련 오류도 내지 않았다. 전체적으로 Polynomial 함수를 사용한 모델이 RBF를 사용한 모델보다 나은 결과를 보였으며 RBF를 사용한 모델은 Type I 오류에 비해 Type II 오류가 매우 많았다. 역전파 신경망 모델은 BPM_old보다 BPM_new가 좀 더 나은 결과를 보였으며, 실제 중요한 Type II 오류는 두 훈련 데이터집합에서 모두 0이 되었다.

4.3 검증 결과

모델들의 예측 정확도를 평가하기 위해 두가지 훈련 데이터 집합으로 훈련시킨 모델들에 두가지 검증 데이터 집합들을 적용하였으며 <표 2>는 그 결과이다.

훈련 2로 훈련시킨 모델에 검증 2를 적용한 결과들은 대부분 모델들에서 거의 오류가 없다. 왜냐하면 훈련 데이터나 검증 데이터 모두 애매성을 없앤 집합이기 때문이다. 훈련 2로 훈련시킨 모델

〈표 2〉 검증 결과

모델 \ 훈련 데이터			검증 1		검증 2	
			Type I	Type II	Type I	Type II
SVM (NotNormalize)	RBF	훈련 1	1/177	9/23	0/177	2/23
		훈련 2	1/177	15/23	0/177	2/23
	Polynomial	훈련 1	1/177	3/23	0/177	1/23
		훈련 2	3/177	14/23	0/177	1/23
SVM (Normalize)	RBF	훈련 1	0/177	15/23	0/177	2/23
		훈련 2	1/177	11/23	0/177	0/23
	Polynomial	훈련 1	0/177	13/23	0/177	2/23
		훈련 2	1/177	12/23	0/177	0/23
BPM_old		훈련 1	4/177	4/23	7/177	0/23
		훈련 2	6/177	11/23	0/177	0/23
BPM_new		훈련 1	2/177	4/23	4/177	0/23
		훈련 2	6/177	11/23	0/177	0/23

에 검증 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. 이 경우에 모든 모델들이 매우 많은 오류를 냄을 볼 수 있으며, 상대적으로 오류가 적어보이는 정규화 데이터에 RBF를 사용한 SVM 모델도 12개의 예측 오류를 내며 이 중 중요한 Type II 오류가 11개를 차지함으로써 예측 모델로서의 의미가 없다.

하지만 반대의 경우인 훈련1로 훈련시킨 모델에 검증 2를 적용시킨 결과는 매우 만족스러운 결과를 보였다. SVM은 데이터 정규화 여부와 커널 함수에 상관없이 Type I 오류는 없었고, Type II 오류도 1~2개에 불과하였다. BPM은 Type I 오류에서 조금 더 안 좋은 결과를 보였다.

가장 일반적인 경우인 훈련 1로 훈련시킨 모델에 검증1을 적용시킨 결과는 SVM 모델에서는 비정규화 데이터와 Polynomial 함수를 사용한 모델이 가장 좋은 결과를 나타내었다. 일반적으로 RBF 함수를 사용한 경우가 Polynomial 함수를 사용한 경우보다 Type II 오류가 많았다. BPM은 BPM_new가 BPM_old보다 Type I 오류 측면에서 근소하게 나았으나 Type II 오류는 모든 결과에서 같게 나옴으로써 비슷한 예측 성능을 보였다.

결과적으로 비정규화 데이터와 Polynomial 커널 함수를 사용한 SVM 모델이 많은 연구들에서 좋은 성능을 낸다고 알려진 BPM보다 약간 나은 성능을 보였다.

5. 결 론

대형 소프트웨어 개발이 많아지고 소프트웨어 프로세스 관리에 관심이 많아짐에 따라 전체 개발 비용을 낮추는데 초기 위험도 예측 모델이 큰 역할을 하고 있다. 발표된 대부분의 모델들은 훈련 데이터 집합이 필요한 VI-SL 모델들로 실제 프로젝트 적용에 어려움이 있었으므로 작은 훈련 데이터 집합으로도 고차원 데이터 공간에서 일반화 성능이 우수하다고 알려진 SVM을 이용한 예측 모델들이 최근에 발표되었다.

본 논문에서는 SDL로 작성한 설계 명세를 SVM을 이용한 모델에 적용하였으며 그 결과 비정규화 데이터와 Polynomial 커널함수를 사용한 SVM 모델이 기존의 오류 역전파 신경망을 사용한 모델보다 약간 우수한 성능을 나타냄을 보였다. 향후 연구는 SVM 모델을 대형 데이터 집합에 적용해보는 것과 아직 품질 예측에 많이 적용되지 않은 베이지안 모델 등을 이용한 예측 모델을 제작, 검증해보는 것이다.

참 고 문 헌

[1] Ebert, C., "Fuzzy classification for software criticality analysis", *Expert Systems with Applications*, Vol.11, No.3(1996), pp.323-342.

[2] Vapnik, V., *The Nature of Statistical Learning Theory*, Springer New York, 1995.

[3] Elish, K. O. and M. O. Elish, "Predicting defect prone software modules using support vector machines", *J. Systems Software*, Vol. 81, No.5(2008), pp.649-660.

[4] Catal, C. and B. Diri, "A systematic review of software fault prediction studies", *Expert Systems with Applications*, Vol.36, No.4(2009), pp.7346-7354.

[5] 홍의석, "소프트웨어 품질 예측 모델을 위한 분류 프레임워크", 「한국콘텐츠학회논문지」, 제10권, 제6호(2010), pp.134-143.

[6] Emam, K. E., W. Melo and J. C. Machado, "The prediction of faulty classes using object oriented design metrics", *J. Systems Software*, Vol.56, No.1(2001), pp.63-75.

[7] Tian, J., A. Nguyen, C. Allen, and R. Appan, "Experience with identifying and characterizing problem-prone modules in telecom-

munication software systems", *J. Systems Software*, Vol.57, No.3(2001), pp.207-215.

[8] Khoshgoftaar, T. M. 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, No.1(1995), pp.85-91.

[9] 홍의석, "훈련데이터 집합을 사용하지 않는 소프트웨어 품질예측 모델," 「정보처리학회논문지」, 제10-D권, 제4호(2003), pp.689-696.

[10] Zhong, S., T. M. Khoshgoftaar, and N. Seliya, "Analyzing Software Measurement Data with Clustering Techniques", *IEEE Intelligent Systems*, Vol.19, No.2(2004), pp. 20-27.

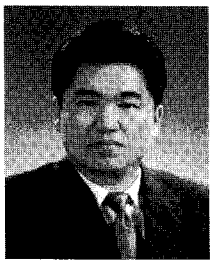
[11] Seliya, N. and T. M. Khoshgoftaar, "Software quality analysis of unlabeled program modules with semisupervised clustering", *IEEE Trans, Systems, Man and Cybernetics*, Vol.37, No.2(2007), pp.201-211.

[12] Xing, F., P. Guo, and M. R. Lyu, "A Novel Method for Early Software Quality Prediction Based on Support Vector Machine", *Proc. International Conference on Software Reliability Engineering*, (2005) pp.213-222.

[13] Gondra, I., "Applying machine learning to software fault-proneness prediction", *J. Systems Software*, Vol.81, No.2(2008), pp.186-195.

[14] 김영미, 정충희, 김현수, "SVM을 이용한 위험모듈 예측", 「정보과학회논문지 : 컴퓨팅의 실제 및 레터」, 제15권, 제6호(2009), pp.435-439.

◈ 저 자 소 개 ◈



홍 의 석 (hes@sungshin.ac.kr)

서울대학교 계산통계학과 전산과학전공에서 1992년과 1994년에 각각 학사, 석사 학위를 취득하였으며 1999년에 서울대학교 전산과학과에서 박사학위를 취득하였다. 1999년부터 2002년까지 안양대학교 디지털미디어학부 교수로 근무하였으며, 2002년부터 현재까지 성신여자대학교 자연과학대학 IT학부 교수로 재직 중이다. 연구 관심분야는 소프트웨어공학 분야 중 소프트웨어 품질, 웹기반 응용 기술 등이다.