학습이론을 이용한 소프트웨어 개발비 예측 모형

박 찬 규*
* 동국대학교 경영학과

Estimating software development cost using machine-learning approach

Chan-Kyoo Park*
* Department of Management, Dongguk University
parkck@dongguk.edu

ABSTRACT

As the portion of information systems(IS) budget to the total government budget becomes greater, the cost estimation of IS development and maintenance projects is recognized as one of the most important problems to be resolved for quantitative and efficient management of IS budget. The primary concern in the cost estimation of IS projects is to estimate software development cost.

In this paper, we propose a new method to estimate software cost using support vector regression(SVR), which has attracted considerable attention because of its good performance and theoretical clearness. The paper is the first study which apply SVR to software cost estimation.

1. 서론

정보시스템이 조직의 운영 효율성과 경쟁력 향상을 위한 필수적인 수단으로 정착됨에 따라 정보시스템을 구축·유지보수·운영하는 비용이 지속적으로 증가하고 있다. 2004년 기준으로 국내 소프트웨어산업 시장규모는 연간 약 180억 달러 이상으로 추정될 정도로 급속히 성장하였다. 정보시스템 비용이 차지하는 비중이 높아짐에 따라, 공공기관 및 일반 기업의 정보화 비용의 효과적 관리가 중요한 문제로 대두되었고 이를 위해서는 정보화 사업의 소요비용을 정확히 예측할 필요가 있다.

정보시스템 구축사업의 경우 전체 사업비용은 크게 소프트웨어개발비와 하드웨어/패키지 소프트웨어 구입비로 구분된다. 하드웨어/패키지 소프트웨어 구입비는 시장가격 조사를 통해 쉽게 예측할 수 있으므로 본 연구에서는 논외로 한다. 소프트웨어개발비는 요구사항의 불확실성과 모호성, 소프트웨어 생산성 측정의 어려움, 과거 데이터의 부족, 신기술에 의한 위험 및 개발환경 의존성 등으로 인해 예측이 쉽지 않다.

소프트웨어 개발비 예측기법에는 크게 알고리즘방법(algorithmic method), 전문가예측방법, 유추(analogy)법, 기계학습(machine learning)에 의한 예측방법 등으로 구분될 수 있다([9][18]). 알고리즘방법으로는 1965년 SDC(System development corporation)모델이 개발된 이후

COCOMO II[6], ISBSG모형[11], ESTIMACS[12], PRICE-S[15], Knowledge PLAN [19] 등 최근까지 수십여 개의 공개 또는 상용 소프트웨어 비용산정 모형들이 개발되었다. 또한, 국내에서도 기능점수를 활용한 알고리즘 기반의 소프트웨어 개발비용 예측에 관한 연구들이 김현수[2], 이양규[이양규], 박찬규[3][4], 김우제[1] 등에 의해 수행된 바 있다. 유추방법은 과거의 유사한 사업으로부터 유추하여 비용을 예측하는 방법이다. 기계학습에 의한 예측방법은 신경망(neural network), 사례기반추론(case-based reasoning), 의사결정나무(decision tree) 등의 기계학습기법 등을 이용하여 소프트웨어 비용을 예측하는 것을 말한다.

기계학습에 의한 소프트웨어 비용예측 방법으로 가장 흔히 사용된 기법은 신경망에 의한 비용예측이다. Shepperd[17]는 문제복잡도, 어플리케이션특이성(novelty of application), 설계도구 사용, 팀크기(team size) 등을 입력요소로 하는 feed-forward 신경망을 통해 소프트웨어 비용을 예측하는 기법을 제안하였다. Kitchenham [13]은 사례기반 추론방법을 도입하여 소프트웨어 비용을 예측하였고, 기존의 회귀분석을 기반으로 하는 알고리즘 방법보다 예측 정확성이 향상됨을 보였다. 또한 나무기반(tree-based) 방법으로 회귀나무(regression tree), 의사결정나무(decision tree) 등을 활용하여 소프트웨어 비용예측을 시도한 연구들도 수행되었다([20][14]). 신경망, 사례기반추론 또는 회귀모형을 이용한 소프트웨어 비용예측의 정확도를 비교하면 신경망에 의한 비용예측이 가장 정확도가 높은 것으로 보고된 바 있다([8]).

본 연구에서는 기계학습 분야에서 새롭게 주목을 받고 있는 SVM(Support Vector Machine)를 소프트웨어 비용예측에 적용하여 그 정확도를 기존의 방법과 비교하고자 한다. SVM은 Vapnik[21]에 의해 제안된 이후 이론적인 명확성과 우수한 성능이 입증되면서 패턴인식, 텍스트 분류, 생체정보학(bioinformatics), 주가 예측, 침입탐지 등 활용 범위를 점차 넓혀가고 있다([7]). SVM은 주로 분류(classification)에 사용되며 SVM를 회귀분석에 사용할 수 있도록 확장한 개념이 SVR(Support Vector Regression)이다. 아직까지 SVR를 소프트웨어 비용예측 분야에 적용한 연구가 없었다는 점에서 본 연구의 첫 번째 의의가 있다. 또한, 국내 상황에 적합한 소프트웨어 비용예측 도구들이 절대적으로 부족한 상황에서 국내 소프트웨어사업에서 수집된 비용자료와 해외에서 널리 통용되고 있는 비용자료를 종합적으로 분석하여 소프트웨어 개발비 예측을 위한 새로운 도구를 제시한다는 면에서도 본 연구의 두 번째 의의를 찾을 수 있다.

이후 전개될 본 논문의 구성은 2절에서 SVM과 SVR를 간단히 소개하고, 3절에서 소프트웨어 비용에 영향을 미치는 비용요소(cost driver)를 알아본다. 4절에서는 전통적인 회귀식을 통한 예측결과와 SVR를 이용한 비용예측 결과를 비교함으로써 SVR이 회귀분석보다 보다 정확한 비용예측 방법이 될 수 있음을 보인다. 마지막으로 5절에서 본 연구의 결론과 추후 연구과제 등을 제시한다.

2. SVM(Suport Vector Machine)과 SVR(Support Vector Regression)

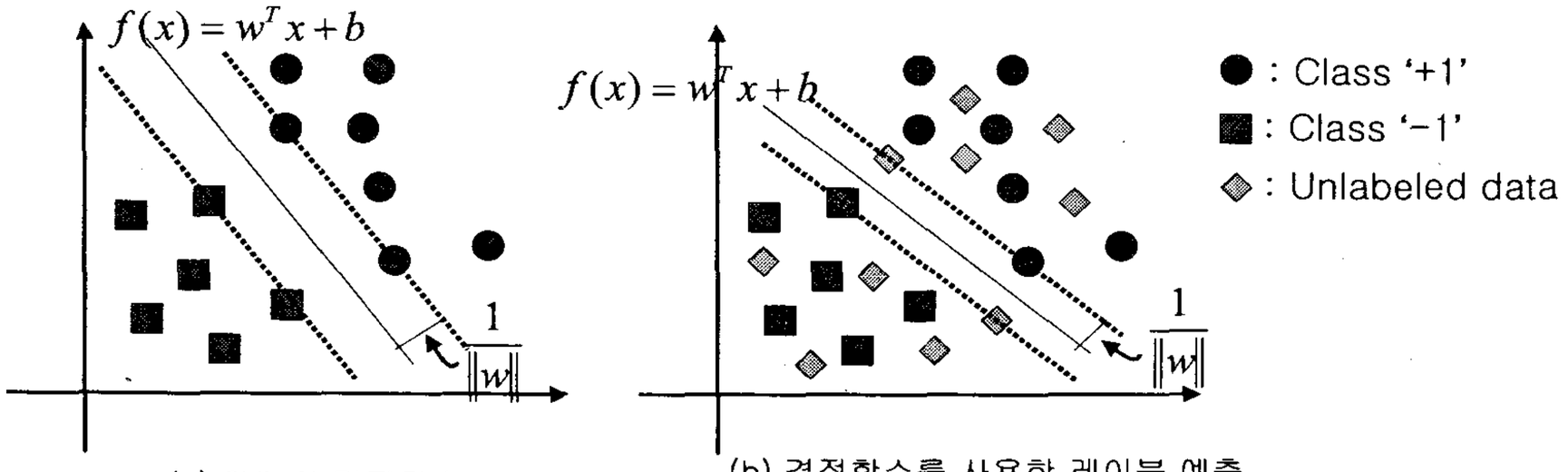
먼저 SVM에 대해 알아보자. 1절에 언급했듯이 SVM은 기계학습에 사용되는 기법이다. SVM은 다차원 상의 점들로 표현되는 학습데이터를 두 개의 그룹으로 분할하는 초평면을 구한 다음 이 초평면을 미지의 데이터가 어떤 그룹에 속할지를 예측하는 결정함수(decision function)로 사용한다. l개의 학습데이터 $(x_1, y_1), \cdots, (x_l, y_l)$가 있고 $x_i \in R^n$, $y_i \in \{1, -1\}$이며 모든 y_i의 값을 알고 있다고 하자. 즉, x_i는 i번째 학습데이터의 다차원 상의 좌표이고 y_i는 i번째 학습데이터가 '+1' 그룹에 속하는지 또는 '-1' 그룹에 속하는지를 나타낸다. [그림 1]의 (a)에서 보는 것처럼 SVM은 학습데이터를 그 레이블 y_i에 따라 두 개의

클래스로 분할하는 초평면(hyperplane) 중에서 두 개의 클래스가 가장 멀리 떨어지도록 분할하는 초평면을 찾는다. 이 때 초평면 $f(x)=w^Tx+b$가 SVM의 결정함수가 되며, 레이블이 미지인 데이터 x에 대해 $f(x)\geq 0$이면 '1' 클래스로, $f(x)<0$이면 '-1' 클래스로 예측한다. $(2/\|w\|)$는 초평면에 의해 분할된 두 클래스간의 거리를 나타내는데 마진(margin)이라 불리는데, SVM은 개념적으로 마진을 최대로 하는 초평면을 결정함수로 사용하게 된다.

$$(SP):\quad \begin{aligned} &\min && w^Tw+C\sum_i^l \xi_i \\ &\text{s.t.} && y_i(w^Tx_i+b)\geq 1-\xi_i,\ \forall i \\ & && \xi_i\geq 0,\ \forall i \end{aligned}$$

$$(SD):\quad \begin{aligned} &\max && e^T\alpha-\frac{1}{2}\alpha^TQ\alpha \\ &\text{s.t.} && y^T\alpha=0, \\ & && 0\leq\alpha_i\leq C,\ \forall i \end{aligned}$$

(SP)의 목적함수는 두 개의 항으로 구성되어 있다. 첫 번째 항은 $w^Tw=\|w\|^2$으로 마진의 역수이다. 결과적으로 $\|w\|^2$을 최소화하는 것은 마진을 최대화하는 w를 찾는 것을 의미한다. 두 번째 항의 $\sum\xi_i$는 경험오류(empirical error)로 구해진 결정함수가 주어진 학습데이터에 대해 얼마만큼 오류(error)를 보이고 있는가를 나타낸다. 여기서 상수 C는 마진과 경험오류간의 상대적 가중치를 나타낸다. 종합해보면 (SP)의 목적함수는 마진을 최대화하고 경험오류를 최소화하는 결정함수 $f(x)$를 찾음을 의미한다.



[그림 1] SVM의 결정함수와 예측

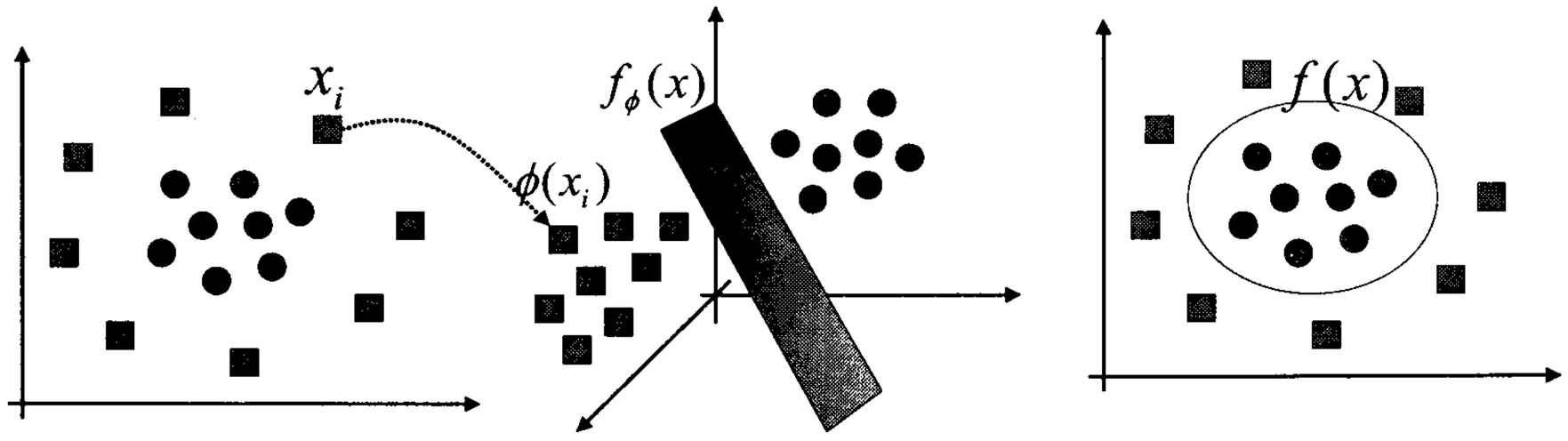
SVM의 결정함수 $f(x)=w^Tx+b$의 w와 b를 구하는 문제를 수리계획법으로 모형화하면 다음 (SP_0)와 같이 이차계획법 문제로 모델링된다.

$$(SP_0):\quad \begin{aligned} &\min && w^Tw \\ &\text{s.t.} && y_i(w^Tx_i+b)\geq 1,\ \text{for}\ i=1,\cdots,l \end{aligned}$$

그러나, 현실 문제에서는 [그림 2]의 (a)처럼 레이블이 다른 학습데이터들이 초평면에 의해 두 그룹으로 완벽하게 분할되지 않고 보통 중첩(overlapping)되는 경우가 많으므로 (SP_0)의 각 제약식을 완화한 모형인 (SP)를 사용한다. (SP)의 쌍대문제는 (SD)와 같은데, 계산상의 여러 가지 장점 때문에 실제로는 (SD)를 푼다([16]). (SD)에서 $e=(1,\cdots,1)^T\in R^l$이고, $Q_{ij}=y_iy_jx_i^Tx_i$이다.

SVM은 학습데이터들을 분할하는 초평면을 결정함수로 사용하는데 [그림 1]에 나타난 바와 같이 결정함수가 단순히 선형함수로만 제약된다면 SVM은 성능 면에서 많은 한계를 가지게 될 것이다. 이러한 문제점을 극복하기 위해 [그림 2]의 (a)와 같이 입력공간(input space)에 있는 학습데이터 x_i를 고차원의 특징공간(feature space)의 점 $\phi(x_i)$로 변환한 다음 (b)와 같이 특징공간의 점들을 그 레이블에 따라 두 개의

그룹으로 분할하는 결정함수 $f_\phi(x)$를 구한다. 특징공간의 결정함수 $f_\phi(x)$을 원래의 입력공간으로 변환하면 (c)와 같이 비선형함수 형태를 띠게 된다.



(a) 입력공간 (b) 특징공간과 결정함수 (c) 입력공간의 결정함수

[그림 2] 입력공간과 특징공간

입력공간의 학습데이터 x_i를 특징공간의 점으로 변환하는 과정은 함수 $\phi: x \rightarrow \phi(x_i)$를 직접 구하지 않고 커널함수(kernel function)를 통해 이루어진다. 특성공간에서 정의되는 SVM의 쌍대문제 (SD)를 풀기 위해서는 $Q_{ij} = \phi(x_i)^T \phi(x_j)$를 계산해야 한다. 만약 $\phi(x_i)$를 명시적으로 구하여 Q_{ij}를 계산하려면 $\phi(x_i)$의 차원 매우 높기 때문에 계산시간이 많이 소모될 것이다. 커널함수는 $\phi(x_i)$와 $\phi(x_j)$를 명시적으로 구하지 않고도 $Q_{ij} = \phi(x_i)^T \phi(x_j)$를 쉽게 계산할 수 있게 해준다. 커널함수로 가장 많이 사용되는 함수는 RBF(Radial Basis Function) 커널, 다항(polynomial) 커널 등이 있는데 RBF 커널은

$$k(x_i, x_j) = \exp\left(-\frac{\| x_i - x_j \|^2}{\xi^2}\right)$$

이고, 다항 커널은

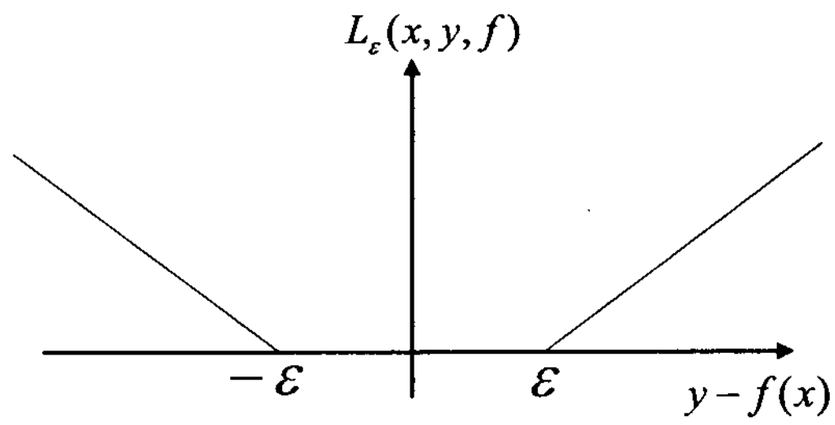
$$k(x_i, x_j) = (1 + x_i^T x_j)^p$$

이다.

SVM이 학습데이터를 '+1' 클래스와 '-1' 클래스로 구분하는데 사용되는데 임의의 실수값을 예측할 수 있도록 SVM를 일반화한 방법은 SVR이다. (SP)의 목적함수에 있는 $\sum_i \xi_i$ 대신에 SVR에서는 다음과 같은 ε-무감도 손실함수(ε-insensitive loss function) $L_\varepsilon(x, y, f)$를 사용한다.

$$\begin{aligned} L_\varepsilon(x, y, f) &= \max(0, |y - f(x)| - \varepsilon) \\ &= \max(0, |y - (w^T x + b)| - \varepsilon) \end{aligned}$$

ε-무감도 손실함수는 실제값 y와 예측된 값 $f(x) = w^T x + b$간의 오차가 ε이하면 $L_\varepsilon(x, y, f) = 0$이고 오차가 ε보다 크면 오차의 절대값에서 ε만큼 차감한 값인 $L_\varepsilon(x, y, f) = |y - f(x)| - \varepsilon$을 갖게 된다. 이를 그림으로 나타내면 다음 그림과 같다.



[그림 3] ε-무감도 손실함수

결국 SVR에서는 실제값 y_i와 예측값 $f(x_i) = w^T x_i + b$의 값을 가능한 한 ε이내로 유지하면서 마진을 최대화하게 되는데 이를 수리계획모형으로 나타내면 다음과 같다([7]).

$$
(RP):\quad
\begin{aligned}
\min\quad & w^T w + C\sum_i^l(\xi_i^+ + \xi_i^-) \\
\text{s.t.}\quad & (w^T x_i + b) - y_i \le \varepsilon + \xi_i^+,\ \forall i \\
& y_i - (w^T x_i + b) \le \varepsilon + \xi_i^-,\ \forall i \\
& \xi_i^+,\ \xi_i^- \ge 0,\ \forall i
\end{aligned}
$$

$$
(RD):\quad
\begin{aligned}
\max\quad & \sum_i y_i(\alpha_i^- - \alpha_i^+) - \varepsilon\sum_i(\alpha_i^- + \alpha_i^+) \\
& -\frac{1}{2}\sum_{i,j}(\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+)x_i^T x_j \\
\text{s.t.}\quad & \sum_i(\alpha_i^- - \alpha_i^+) = 0, \\
& 0 \le \alpha_i^+,\ \alpha_i^- \le C, \forall i
\end{aligned}
$$

SVR도 SVM에서와 마찬가지로 커널함수를 써서 학습데이터를 특징공간의 점으로 변화시킨 다음 특징공간에서 학습을 수행하게 되는데 이때 특징공간에서의 (RD)에 나타나는 $Q_{ij} = \phi(x_i)^T\phi(x_j)$ 값은 커널함수 $k(x_i, x_j)$를 사용하여 계산하게 된다.

3. 국내외 소프트웨어 개발비 예측 모형

SVR을 이용하여 소프트웨어 개발비를 예측하기 위해서는 먼저 소프트웨어 개발비용에 영향을 주는 요인들을 식별해야 한다. 이를 위해 기존에 개발된 여러 가지 알고리즘 기반의 예측모형을 분석하여 소프트웨어 개발의 비용요소를 추출해 본다.

일반적으로 알고리즘 기반의 예측모형은 먼저 개발하고자 하는 소프트웨어의 규모를 예측하고, 규모와 시스템의 특성이나 프로젝트 환경 등을 반영하는 보정요소를 고려하여 소프트웨어 비용을 예측하게 된다. 소프트웨어 규모는 LOC(Line of Code) 또는 기능점수(function point) 방식을 가장 널리 사용하고 있는데, 예측의 편이성, 측정결과의 정확성 및 일관성 측면에서 기능점수 방식이 우수하여 본 연구에서는 기능점수 방식의 소프트웨어 규모측정만을 고려하기로 한다. 기능점수 방식에 관한 상세한 설명은 [10]를 참조하기 바란다.

1965년 미국 SDC(Systems Development Corporation)사에서 169개의 소프트웨어 프로젝트를 분석하여 104개의 비용보정요소 중 통계적으로 의미가 있는 14개의 보정요소를 갖는 소프트웨어 비용예측모델을 개발한 후 현재까지 많은 비용예측모델이 개발·개선되어 왔다. 그 중에서

[표 1] 여러 가지 모형의 보정요소

모 형	개 발 자	주요 보정요소
COCOMO	B. W. Boehm	소프트웨어 신뢰성, DB크기, 제품복잡도, 실행시간제약, 주기억장치제약, 가상기계가변성, 반응시간, 분석능력, 어플리케이션경험, 프로그래머능력, 가상기계경험, 개발언어경험, 최신프로그래밍기술사용 정도, 소프트웨어 도구사용 정도, 개발일정
IFPUG	IFPUG	데이터 통신, 분산처리, 성능, 하드웨어 제약, 처리율, 온라인 데이터입력, 최종사용자 용이성, 온라인 갱신, 처리복잡도, 재사용성, 설치용이성, 운영용이성, 복수사이트, 변경용이성
ISBSG	ISBSG	플랫폼(메인프레임, 중간, PC), 개발언어(3GL, 4GL, ApG),
PRICE-S	PRICE Systems	응용도메인(상용, 군용) 통합수준, 개발언어, 생산성요소, 복잡도
Knowledge PLAN	Caper Jones	플랫폼, 응용도메인, 복잡도, 개발언어, 인력구성, 사용기술, 소프트웨어 도구, 개발환경, 보안/성능 요구수준, 문서화수준, 테스트 복잡도

가장 널리 사용되고 있는 공개모형인

COCOMO/COCOMO II모형[6], IFPUG모형[10], ISBSG모형[11]과 상용모델인 PRICE-S[15], KnowledgePLAN[19]에 포함된 보정요소 간략하게 정리하면 [표 1]과 같다. [표 1]에서 IFPUG모형은 원래는 기능점수를 산정하기 위한 방법이지만, 시스템 일반특성에 따라 기능점수를 보정하는 과정에서 고려하는 시스템 일반특성 14 가지가 다른 모형의 보정요소와 같다고 볼 수 있다.

[표 1]에 포함된 모형 중에서 공개된 모형은 COCOMO와 ISBSG 모형이다. IFPUG 모형은 공개되어 있긴 하지만 시스템 특성을 반영한 기능점수 규모 산정 과정까지만 제시할 뿐 비용을 추정하는 모형은 제시하지 않고 있다. COCOMO 모형의 비용예측 공식은 다음과 같다.

$$PM = a \times S^{b} \times \prod_{i=1}^{15} f_i \qquad \text{(식 3.1)}$$

PM는 필요인력(person-month)을 나타내고, S는 소프트웨어 규모이고, a, b는 수준(level)과 개발유형에 따라 정해지는 상수이다. 또한, f_i는 노력승수(effort multiplier)라고 하는데, 프로젝트의 특성과 여러 가지 보정요소를 고려하여 결정되는 값이다. COCOMO에서 사용하는 보정요소에는 15가지가 있으며 각 보정요소의 복잡도를 매우 낮음(very low), 낮음(low), 보통(nominal), 높음(high), 매우 높음(very high), 극히 높음(extra high) 등 6단계로 구분하여 가중치를 주고 있다. 예를 들어, 요구되는 소프트웨어 신뢰성이 매우 낮은 경우 0.75에서부터 매우 높은 경우 1.40까지의 값을 갖고 있다.

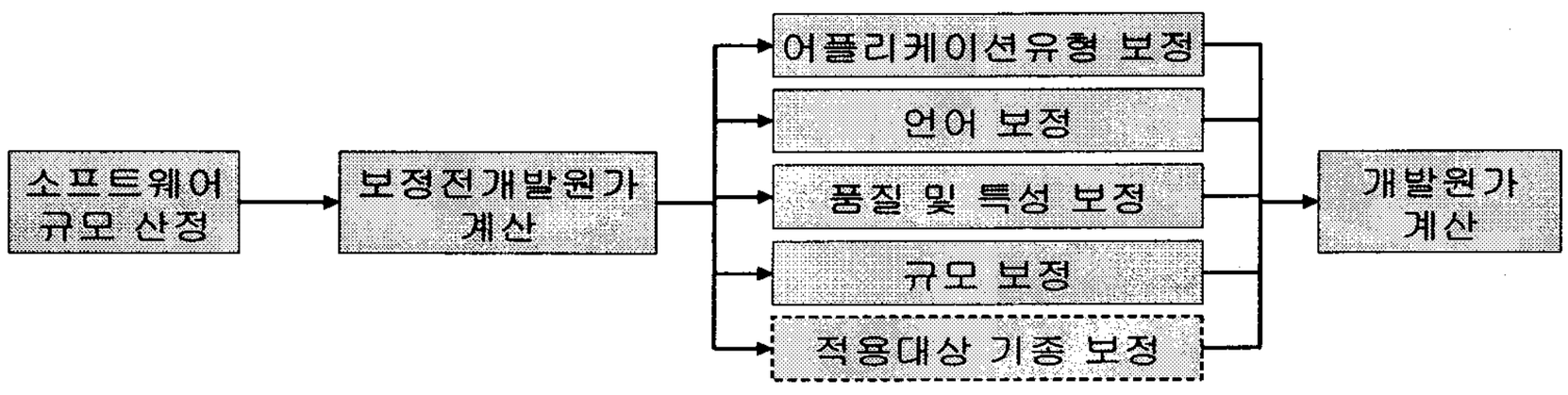
ISBSG모형은 ISBSG에 수집한 약 731개의 소프트웨어 비용자료를 분석하여 제시한 회귀모형으로서 프로젝트 인도율(delivery rates), 필요인력(effort), 프로젝트기간(duration) 등을 예측할 수 있다. 필요인력을 예측하는 식은 다음과 같다([11]).

$$PWE = C \times S^{E_1} \times MxTeam^{E_2} \qquad \text{(식 3.2)}$$

$$PWE = C \times S^{E} \qquad \text{(식 3.3)}$$

(식 3.2)는 소프트웨어의 기능점수 규모와 개발팀의 최대 크기를 알고 있을 경우 개발비를 예측하는 식이고 (식 3.3)은 기능점수 규모만 알고 있을 경우 개발비를 예측하는 식이다. (식 3.2)와 (식 3.3)에 포함된 상수 C, E, E_1, E_2 등의 값은 보정요소에 따라 달라진다. 예를 들어 (식 3.3)에서 플랫폼이 PC이고 4GL를 사용하는 경우 $C = 3.38$, $E = 0.974$를 사용하여 개발비를 예측하게 된다.

다양한 비용예측 모형들이 개발·활용되고 있는 해외와는 달리 국내에서는 정보통신부에서 제정·고시한 소프트웨어사업대가기준을 통일적으로 사용해 왔다. 소프트웨어사업대가기준은 1989년 제정된 이래 공공기관에 발주하는 소프트웨어 사업의 원가와 대가를 산정하는 기준으로 활용되어 왔으며, 소프트웨어사업에 관한 전문지식 없이도 소프트웨어 사업의 예산을 산정할 수 있는 통일적 기준을 제시함으로써 소프트웨어사업의 발주와 관리를 용이하게 하는데 기여하였다. 그러나 산정결과의 정확성면에서 꾸준한 문제점이 제기되고 있어 지속적인 보완이 필요한 실정이다. 소프트웨어사업대가기준에 따른 소프트웨어 개발비 예측 절차는 다음 [그림 4]와 같다.



[그림 4] 소프트웨어사업대가기준의 개발비 산정 절차

기능점수방식으로 개발 대상 소프트웨어의 규모를 산정한 다음 기능점수 단가를 곱하여 보

[표 2] ISBSG 및 국내 비용자료의 수집항목

구분	주요 수집항목
ISBSG	프로젝트 ID, 규모산정 방법, **기능점수**, **기능점수보정인자**, **기능점수측정방식**, **소요노력(총괄)**, 소요노력(측정수준), 데이터품질, 최대팀 크기, **개발유형**, **개발플랫폼**, **언어유형**, 주프로그래밍언어, 사용 DBMS, CASE 사용여부, 방법론사용여부, 개발방법, **사업기간**, 사업중단기간, **구현일자**, 불량개수, 사용자수(사업, 위치, 동시사용자 기준), 조직 유형, 업무영역 유형, **어플리케이션 유형**, 패키지사용여부, **사업범위**
국내 비용자료	사업번호, **사업연도**, 사업비(예산, 계약, 개발비, 장비도입비 등), **사업기간**, **사업범위**, **어플리케이션 유형**, 프로그램본수, **기능점수**, **기능점수보정인자**, **개발언어**, 입력화면수, 보고서수, 출력화면수, **적용대상기종**, **투입인력**

정전 개발원가를 계산한 다음 이를 보정하여 개발원가를 계산한다. [그림 4]에서 알 수 있듯이 소프트웨어사업대가기준에서는 여러 가지 보정요소를 포함하고 있다. 어플리케이션 유형, 언어, 적용대상 기종, 규모 보정은 2004년 이전의 소프트웨어사업대가기준에 포함된 보정요소이고, 2004년에 개정된 사업대가기준에는 적용대상 기종 보정이 삭제되고 품질 및 특성 보정요소로 통합되었다. (식 3.1) ~ (식 3.3)의 예측모형은 규모가 증가함에 따라 필요인력이 지수적으로 증가하지만, 소프트웨어사업대가기준의 기능점수 단가는 규모에 상관없이 일정하다. 따라서 이러한 문제점을 해결할 수 있도록 포함된 보정요소가 규모보정이다.

4. 비용요소 선정

3절에서 알아보았듯이 소프트웨어 개발비 예측을 위해서는 먼저 어떤 비용요소를 고려할 것인가를 정해야한다. 비용요소 선정 시에는 개발비와의 관련성뿐만 아니라 데이터의 획득가능성, 다른 비용요소와의 상관관계 등이 종합적으로 고려되어야 한다. 본 연구에서는 해외 소프트웨어 개발사업 비용자료로 ISBSG 데이터와 국내 소프트웨어 개발사업 비용자료로 한국전산원에서 구축한 데이터(이하 '국내 비용자료'라 부른다)를 활용한다. 먼저 데이터의 획득가능성을 확인하기 위해 ISBSG 데이터와 국내 비용자료의 주요 수집항목을 정리하면 [표 2]와 같다. 비용과 전혀 관련이 없는 수집자료는 [표 2]에서 제외하였다.

[표 2]에서 볼 수 있듯이 ISBSG는 비용에 영향을 미칠 가능성이 있는 다양한 자료들을 수집하고 있지만 회원들의 자발적인 참여로 수집된 자료이기 때문에 많은 결측치(missing value)를 포함하고 있다. 반면에 국내 비용자료에는 사업결과물로부터 획득 가능한 제한된 항목들만이 포함되어 있지만, 결측치를 갖는 사업은 매우 적다. ISBSG 데이터와 국내 비용자료에 공통으로 수집되는 항목은 [표 2]에 굵게 밑줄 쳐진 항목들이다. 공통항목들에 대해 알아보고 이들을 어떻게 예측모형에 반영할 수 있는지 알아본다.

① 소프트웨어 규모

먼저 소프트웨어 규모는 ISBSG와 국내 비용자료 모두 기능점수 방식에 의해 계산되었다. 국내비용자료는 모두 IFPUG 측정규칙에 따라 기능점수 크기를 측정하였다. 반면, ISBSG는 IFPUG 측정규칙뿐만 아니라 NESA, MARK II 등 여러 가지 다른 기능점수 규칙에 따라 측정된 데이터들도 포함되어 있다. 국내 비용자료와의 호환을 위해 IFPUG 규칙 이외의 다른 기능점수 규칙에 의해 소프트웨어 규모를 측정한 데이터를 제외하기로 한다. 또한, 국내비용자료에는 보정전(unadjusted) 기능점수와 기능점수 보

정인자(adjustment factor)가 모두 조사되어 있으나, ISBSG에는 기능점수 보정인자가 없이 단지 보정후 기능점수만 포함된 자료들이 많다. 따라서 보다 많은 데이터를 분석에 포함할 수 있도록 보정후 기능점수를 소프트웨어 규모로 사용하기로 한다.

② 소요인력

ISBSG의 소요노력(work effort)은 투입된 인력의 총시간으로 표시되어 있으나, 국내비용자료의 투입인력은 인-월(man-month)로 표시되어 있다. 1개월을 4주로 보고 주 44시간 근무를 기준으로 인-월 단위의 투입인력을 투입시간으로 변환하여 사용한다.

③ 개발플랫폼

ISBSG의 개발플랫폼(development platform)은 대상사업의 하드웨어 유형을 PC, 중형(mid-range), 대형(mainframe) 등 3 가지로 분류한다. 반면에 소프트웨어사업대가기준에 따라 국내비용자료의 적용대상 기종은 PC, 워크스테이션(client-server, UNIX 시스템 포함), 중대형 등 3 가지로 분류되어 ISBSG의 분류기준과는 약간 다르다. 따라서 ISBSG의 개발플랫폼과 국내비용자료의 적용대상 기종을 종합하여 개발플랫폼을 PC와 중대형으로 2 가지로 분류하기로 한다. 중대형은 대부분 UNIX 계열의 client-server 구조를 기반으로 하므로 PC 기반 플랫폼과 구분되어 비용에 미치는 영향이 다를 수 있음을 반영하기도 한다.

④ 언어

ISBSG의 언어유형은 2GL, 3GL, 4GL, AP(Application Generator) 등 4 가지로 분류되고, 주프로그래밍언어 항목은 프로젝트에 주로 사용된 언어를 의미한다. 그러나 대부분의 사업에서 주프로그래밍언어 항목이 빠진 채 언어유형만 표시되어 있다. 국내비용자료의 개발언어에는 사업에 사용된 언어가 모두 표기되어 있으나, 전체 소프트웨어 중 어느 정도 비율로 각 언어가 사용되었는지는 알 수 없는 실정이다. 따라서 사용언어를 ISBSG와 같이 2GL, 3GL, 4GL, AP로 구분하고 국내사업도 사용언어에 따라 재분류한다.

⑤ 사업기간과 사업연도

사업기간은 ISBSG와 국내비용자료 모두 개월 수로 표시되어 있다. 사업범위는 ISBSG와 국내비용자료 모두 소프트웨어개발기간의 어느 단계에 해당하는 사업인가를 나타내는데 요구사항 분석에서 시스템 개발, 시험, 설치까지 전단계를 포함하는 사업만을 분석대상으로 한다. 또한 ISBSG의 구현일자와 국내비용자료의 사업기간으로부터 소프트웨어 개발연도를 유추해 낼 수 있다. 시간에 지남에 따라 소프트웨어 생산성이 변화할 수 있고 이로 인해 소요인력의 변화 여부를 검증하기 위해 사업수행연도를 예측모형에 포함하기로 한다. 1980년을 0으로 하여 1년 단위로 사업수행연도를 표시하여 분석한다.

⑥ 어플리케이션 유형

ISBSG에서는 어플리케이션 유형을 MIS, 업무/생산시스템, 사무정보시스템, 의사결정지원, 공정제어 등 10 가지 이상 다양한 형태로 분류하고 있다. 반면 국내 비용자료는 소프트웨어사업대가기준에 따라 업무처리용, 과학기술용, 멀티미디어용, 지능정보용, 시스템용, 통신제어용, 공정제어용, 지휘통제용 등 8 가지로 분류하고 있다. ISBSG의 분류와 국내 비용자료의 분류가 일치하는 부분은 그대로 사용하고 서로 다르게 분류된 부분은 재분류하여 다음 [표 5]과 같이 어플리케이션 유형을 구분하기로 한다. 표에서 지휘통제용, 과학기술용 등 ISBSG 및 국내비용자료에 나타나지 않은 어플리케이션 유형은 제외하였다.

⑦ 조직 유형

발주기관에 따른 소프트웨어개발비 영향여부를 파악하기 위해 조직유형에 관한 비용요소를 도입한다. ISBSG에는 발주기관 조직유형을 보험, 은행, 금융업, 정부 및 공공기관, 제조업, 도소매업, 컴퓨터, 운송업, 통신업, 전기/가스/수도 등

여러 가지 유형으로 분류하고 있다. 이를 유사한 몇 개의 그룹으로 재분류할 필요가 있는데, 본 연구에서는 보험/은행/금융업, 공공기관/공공재 분야, 제조업, 컴퓨터 및 통신업, 기타 등 4 가지 분야로 조직유형을 분류하여 조직유형간 개발비 차이 유무를 분석한다. 국내 비용자료는 모두 공공기관으로 조직유형을 분류하였다.

마지막으로 ISBSG의 개발유형(development type)은 프로젝트가 신규개발, 기능개선, 재개발인가를 의미하는데 국내 비용자료는 모두 신규개발 사업을 대상으로 수집된 자료이다. 따라서 ISBSG의 데이터 중 개발유형이 기능개선이거나 재개발인 사업은 모두 제외하고 신규개발인 자료만을 분석대상으로 한다.

5. 국내외 비용자료 적용결과

본 절에서는 4절에서 선정된 비용요소를 독립변수로 하여 소프트웨어개발비용을 예측한 결과를 제시한다. 먼저 불완전한 데이터 및 이상치(outlier)를 갖는 데이터 제거 과정을 설명하고 최종적으로 분석대상으로 남은 데이터에 대한 분포를 설명한다. 또한, SVR 적용 결과를 제시하기 이전에 SVR에 의한 예측 정확성을 비교해 볼 수 있는 기준으로 선형회귀모형에 의한 예측결과를 제시하고, 마지막으로 SVR 적용결과를 제시한다.

적용데이터

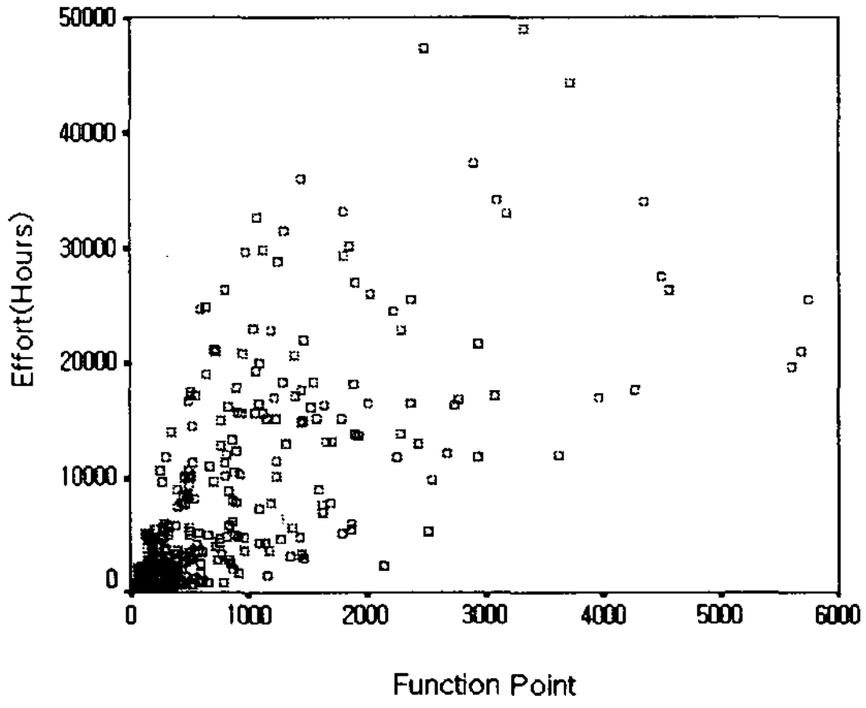
4절에서 비용요소로 선정된 속성에 관한 측정값이 없는 자료, 소프트웨어 개발 프로젝트로 보기 어려운 사업 등 분석 대상에 포함되기 부적할한 자료를 제거하였는데, ISBSG와 국내비용자료에서 제거된 데이터들과 제거 기준을 요약하면 [표 3]과 같다. 총 1,238개의 ISBSG 데이터 중 908개가 제거되었고 남은 데이터는 332개이고, 총 76개의 국내비용자료에서 10개가 제거되고 남은 데이터는 66개이다. [표 3]에 의해 제거되고 남은 자료는 총 398개이다.

[표 3] 제거 기준 및 제거 사업 개수

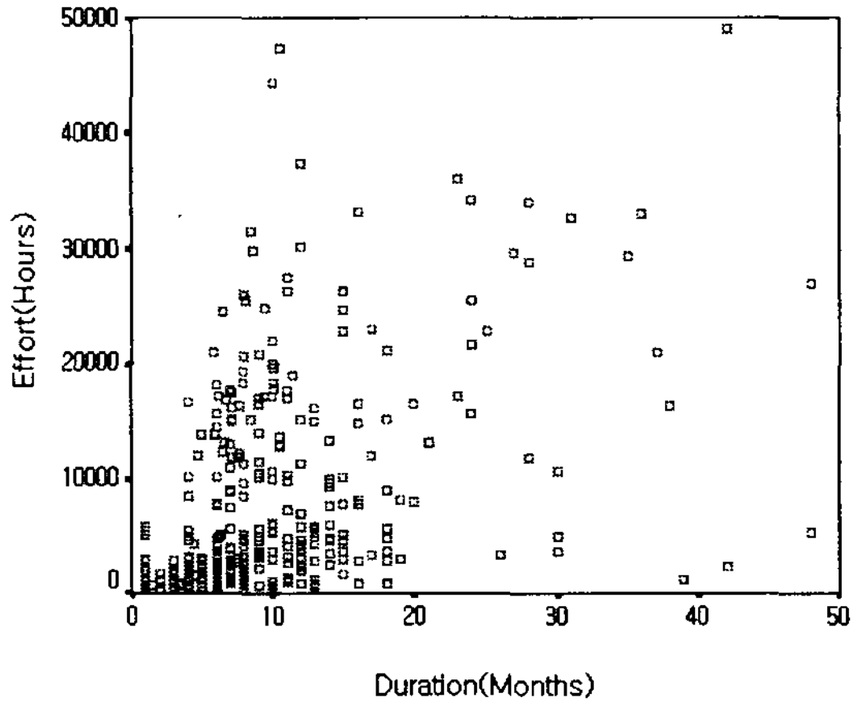
ISBSG 데이터		국내비용자료	
제거 기준	제거 사업수	제거 기준	제거 사업수
IFPUG 이외의 규모산정	120개	DB구축 인력 비율이 높은 사업	3개
기능점수 크기 부재	7개		
기능개선, 재개발, 패키지 구매	647개		
개발 플랫폼, 언어 유형 결측 자료	38개	투입인력자료 부재	6개
사업기간 결측 자료	61개		
어플리케이션 유형, 조직 유형, 사업 분야 유형 결측자료	32개	동일 사업이나 기능점수 측정범위에 따라 2개로 구분된 사업	1개
언어유형인 2GL인 자료	1개		
사업시작연도 부재	10개		
합 계	**916개**	**합 계**	**10개**

다음으로 이상치에 의해 예측 결과가 왜곡되는 것을 방지하기 위해 다른 데이터에 비해 매우 작거나 큰 속성값을 갖는 자료를 제거하기로 한다. 첫 번째로 기능점수 크기가 매우 작거나 또는 매우 큰 사업들을 제외한다. [표 3]의 결과로 남은 총 398개의 데이터의 기능점수 크기 분포는 11FP~26,968FP이다. 이 중 기능점수 크기가 50FP 이하인 사업과 6,000FP 이상인 사업 13개를 제외한다. 두 번째로 기능점수 당 투입노력 비율이 1Hour/FP로 매우 작은 사업과 45Hours/FP로 매우 큰 사업 22개를 제외하였다. 마지막으로 투입시간이 60,000Hours로 다른 데이터에 비해 현격히 많은 사업 6개를 제외하였다. 따라서 ISBSG 데이터와 국내비용자료 중에서 불완전한 자료와 이상치를 제외하고 분석 대상으로 선정된 데이터는 총 347개이다. 이들 데

이터에 대해 기능점수크기와 투입인력, 사업기간과 투입인력의 분포를 그래프로 나타내면 [그림 5], [그림 6]과 같다.



[그림 5] 기능점수와 투입인력



[그림 6] 사업기간과 투입인력

선형회귀모형에 의한 비용예측

3절에서 선정된 비용요소들을 고려하여 (식 5.1)과 같이 회귀모형을 구성하였다. 예측할 종속변수 E는 인력의 투입시간이고, S는 소프트웨어의 기능점수 규모이다. T는 사업기간, Y는 사업수행연도, P는 개발플랫폼, L는 개발언어, A는 어플리케이션 유형, O는 조직유형, K는 ISBSG 데이터와 국내비용자료간의 생산성 차이를 반영하기 위해 도입한 변수이다. 소문자로 표시된 a, b, c, d는 회귀모형에 의해 도출되는 계수들이다.

$$E = a \times S^b \times T^c \times Y^d \times P \times L \times A \times O \times K \qquad \text{(식 5.1)}$$

(식 5.1)은 네 가지 가정을 토대로 도출되었다. 첫 번째 가정은 투입인력은 소프트웨어 규모가 증가함에 따라 지수적으로 증가한다는 것이다. 이 가정은 기존의 대부분의 소프트웨어 개발비용 예측모형에서 사용되고 있다. 두 번째 가정은 소프트웨어 개발기간 단축에 따라 필요한 투입인력은 지수적으로 증가한다는 것이다. 이 가정 또한 SLIM, ISBSG 등 대부분의 예측모형에 통용되고 있다. 세 번째 가정은 소프트웨어 개발생산성은 개발연도가 최근에 가까울수록 지수적으로 증가한다는 것이다. 이는 소프트웨어 개발생산성이 매년 균일한 비율로 개선됨을 의미하는데, 소프트웨어 개발생산성이 시간이 지남에 따라 개선되기는 하지만 균일한 비율로 개선되지는 않기 때문에 현실적으로 성립되기 어려운 가정이다. 그러나 소프트웨어 개발생산성의 개선비율을 매년 다르게 하여 회귀모형을 구성하는 것은 이론적으로나 계산복잡도면에서 간단치 않다. 따라서 개선비율의 기하평균을 선형회귀모형에 반영시키기 위해서는 세 번째 가정이 필요하다. 네 번째 가정은 개발플랫폼, 언어유형, 어플리케이션유형, 조직유형, 등이 달라짐에 따라 일정 비율만큼 투입인력에 영향을 준다는 가정이다. 이는 COCOMO 모형, 소프트웨어사업대가기준 등에서 사용되고 있는 가정이다. 마지막으로 비용자료의 국내외 구분에 따라 일정 비율만큼 투입인력이 달라진다는 가정이다. 이는 국내 소프트웨어사업과 해외 소프트웨어 사업간의 생산성의 차이를 반영하기 위한 가정이다.

(식 5.1)을 선형회귀식으로 변환하기 위해 양변에 로그(log)를 취하였다. 이후 개발플랫폼, 언어유형, 어플리케이션유형, 조직유형, 국내외 구분 등의 변수는 0 또는 1의 값을 갖는 더미변수

(dummy variable)를 도입하여 변환하였다. 예를 들면, 개발언어의 경우 3GL, 4GL, AP 등 3 개의 언어 유형을 고려하므로 더미변수 $LANG_4GL$와 $LANG_AP$를 도입하여 언어 유형이 4GL이면 $LANG\_GL=1$, $LANG_AP=0$으로 나타낸다. 언어유형이 3GL이면 두 더미변수 모두 0을 갖도록 표현하였다.

회귀모형의 계수를 구하기 이전에 독립변수들간의 상관성을 검토한 결과 조직유형과 국내외 구분간의 상관성, 기능점수와 사업기간간의 상관성이 매우 높은 것으로 나타났다. 이는 국내 비용자료가 전부 공공기관을 상대로 수집되어서 총 분석대상 데이터 347개 중에서 90개의 자료가 공공기관의 프로젝트이고 그중 58개가 국내 비용자료이기 때문이다. 또한 회귀식의 계수를 구해 본 결과 조직유형과 사업기간은 투입시간에 통계적으로 영향을 미친다고 볼 수 없는 것으로 나타나 조직유형과 사업기간은 회귀모형에서 제외하기로 한다. 조직유형을 제거한 후 회귀식의 계수를 다시 구해본 결과 어플리케이션 유형과 사업수행연도의 계수가 모두 통계적으로 0이 아니라고 볼 수 없는 것으로 나타나 어플리케이션 유형과 사업수행연도도 회귀모형에서 제거하기로 한다. 결과적으로 다음과 같은 선형회귀모형과 회귀계수를 얻을 수 있었다. (식 5.2)의 회귀모형에서 $R^2=0.674$ 이다.

$$\log E = a + b\log S + c\log P + d\log L + e\log K \qquad \text{(식 5.2)}$$

[표 4] 회귀식의 계수

회귀식 계수		계수값	유의확률
a		1.237	0.000
b		0.827	0.000
c	PC일 경우	0	-
	중대형일 경우	0.269	0.000
d	3GL일 경우	0	-
	4GL일 경우	-0.158	0.000
	AP일 경우	-0.249	0.002
e	해외사업의 경우	0	-
	국내사업의 경우	0.219	0.000

SVR 적용 결과

(식 5.1)에 쓰인 독립변수를 모두 사용하고 투입시간을 종속변수로 하여 SVR로 학습을 수행하였다. 학습프로그램으로는 LIBSVM을 사용하였으며, 커널함수로는 RBF를 사용하였고 이때 $\xi^2=2/3$, $\varepsilon=0.3$으로 설정하였다. SVR를 통한 예측결과와 (식 5.2)의 선형회귀분석을 통한 예측결과를 비교해보면 [표 8]과 같이 정리할 수 있다.

[표 5] 선형회귀모형과 SVR의 예측 정확성 비교

비교항목	선형회귀모형	SVR
R^2	0.674	**0.832**
Mean Squared Error	0.319	**0.298**
Pred(0.05)	140	**224**
Pred(0.10)	251	**310**
Pred(0.20)	336	**341**

[표 8]에서 Pred(0.05), Pred(0.10), Pred(0.20)는 예측오차가 각각 5%, 10%, 20% 이내인 사업의 개수를 말한다. 선형회귀모형과 비교하여 SVR이 모든 비교항목면에서 보다 예측 정확성이 높은 것으로 나타났다.

6. 결론

본 연구는 해외에 널리 사용되는 소프트웨어 개발 사업 비용자료와 국내 비용자료를 종합적으로 분석하여 소프트웨어 개발비용을 예측하는 모형을 제시하였다. 본 연구에서 고려한 Support Vector Regression은 방법은 선형회귀분석을 이용한 방법보다 예측 정확성면에서 우수한 결과를 보였다.

본 연구는 데이터 획득의 한계로 인해 소프트웨어 개발 비용에 영향을 미치는 요소 중에 일부밖에 고려하지 못하였는데, 추후 다양한 보