プロジェクト類似性に基づいた線形重回帰モデルによるソフトウェア開発労力の予測

肖 霄[1,a)]　土肥 正[1,b)]

概要：本論文では，現行のソフトウェア開発プロジェクトの開発労力を予測するために，プロジェクト類似性に基づいた線形重回帰モデルを提案する．具体的には，現行プロジェクトと過去プロジェクトの類似度を算出し，類似プロジェクトのみを線形重回帰モデルの入力として与える．数値例では，実際のソフトウェア開発プロジェクトデータを用いて，提案モデルの有効性について定量的な評価を行う．結果として，提案モデルは従来の回帰モデルである最小二乗回帰，Ridge 回帰，Lasso 回帰よりも高い予測精度を持つことが示される．

キーワード：ソフトウェア開発労力，プロジェクト類似性，線形重回帰，Ridge 回帰，Lasso 回帰，予測．

Prediction of Software Development Effort via Project Analogy-based Linear Multivariate Regression Models

Xiao Xiao[1,a)]　Tadashi Dohi[1,b)]

Abstract: In this paper, we propose project analogy-based linear multivariate regression models to predict the necessary development effort of the current progressing software project. More specifically, we identify the similar projects by calculating the similarity between target software development project and completed ones, and utilize only the similar projects as the input of the linear multivariate regression models. In numerical studies with real software development project datasets, we evaluate the effectiveness of our models quantitatively. The analysis results show that our models achieve higher prediction accuracy than the conventional regression-based models without similarity measures, such as OLS regression, Ridge regression and Lasso regression.

Keywords: software development effort, project analogy, linear multivariate regression, Ridge regression, Lasso regression, prediction.

1. はじめに

ソフトウェア開発労力予測はソフトウェア開発プロジェクト管理の成功を達成する上で重要な課題となる．過去 40 年に渡り，COCOMO [1], COCOMO II [2], SLIM [3], ファンクションポイント分析 [4] など，開発労力を予測するための手法が数多く提案されている．これらは専門家による経験的な評価技法 [5] とは違い，例えば COCOMO [1] のように開発労力と開発規模の関係が数学的に定式化されており，開発規模を入力情報として与えるだけで容易に開発労力が予測出来るため，多くの実証分析において利用されてきた．しかし，ソフトウェアの開発プロセスや開発手法は急速に多様化しており，これらの基本的なモデルだけでは高精度な予測を行うことが困難になってきている [6]．このような背景から，より汎用的でかつ客観的な評価が行えるデータマイニング技術 [7], [8] がソフトウェア開発労力の予測に適用されるようになった．最近，Dejaeger [8] らは 9 つの実データを用いて既存予測手法の比較を行った．

[1] 広島大学工学研究院
1-4-1, Kagamiyama, Higashi-hiroshima, Hiroshima 739-8527, Japan
[a)] xiaoxiao@rel.hiroshima-u.ac.jp
[b)] dohi@rel.hiroshima-u.ac.jp

彼らが着目したのは，9 種類の回帰モデルに加え，ニューラルネットワーク [9] やサポートベクトルマシン [10] などを含めた 13 種類の技法であり，結果として，データの対数変換を伴う最小二乗法 (ordinary least squares; OLS) による線形重回帰モデルが最もよい予測精度を持つことが示された．

一方，近年では，ソフトウェア開発プロジェクト間の類似性に着目した予測手法が特に注目されており，事例ベース推論 (case-based reasoning; CBR) [11], [12] や協調フィルタリング (collaborative filtering; CF) [13], [14] に基づいた方法が提案されている．過去プロジェクトから現行プロジェクトに類似したデータを抽出するため，プロジェクトの個別性をより強く反映した予測を行えることが特徴となっている．ソフトウェア開発現場では頻繁に専門家による経験的な評価技法 [5] が用いられることもあるが，CBR や CF はこのような経験に頼った予測を系統的に扱う方法である．本論文では，現行プロジェクトと過去プロジェクトの類似性を従来の回帰モデルに取り入れた予測手法を提案する．まず，第 2 節と第 3 節では，従来の線形重回帰モデルと CBR による予測手法について紹介する．第 4 節では，プロジェクト類似性に基づいた線形重回帰モデルを構築する．第 5 節では，2 種類のソフトウェア開発プロジェクトデータを用いて，提案モデルの有効性について定量的な評価を行う．第 6 節では本論文のまとめと今後の課題について述べる．

2. 線形重回帰による予測手法

線形重回帰 (linear multivariate regression; LMR) モデルでは，ソフトウェア開発労力を目的変数とし，それを複数のプロジェクト特性の線形結合によって表現する．一般的に，LMR モデルは次式の形式を持つ．

$$Y_i = \sum_{j=0}^{n} \beta_j X_{i,j} + \varepsilon. \tag{1}$$

Y_i と $X_{i,j}$ はそれぞれプロジェクト $i\ (=1,\ 2,\ldots,\ m)$ のソフトウェア開発労力とプロジェクト特性 $j\ (=0,\ 1,\ldots,\ n)$ を表す．ここで，一般性を失うことなく，$X_{i,0}=1$ と仮定する．また，$\beta_0,\ \beta_1,\ldots,\ \beta_n$ は回帰係数であり，確率誤差項 ε は式 (1) で与えられる線形性と観測値とのズレを解釈する確率変数である．

2.1 最小二乗法

最小二乗法 (ordinary least squares; OLS) は満足の出来る予測精度と適用の簡易さを特徴に持つため，従来の LMR モデルの中でも中心的な役割を担ってきた．まず，以下の仮定を設ける．

(仮定 1) 観測値の誤差には偏りがない．すなわち，誤差の平均が 0 である．

$$\mathrm{E}[\varepsilon] = 0.$$

(仮定 2) 観測値の誤差の分散は一定である．

$$\mathrm{Var}[\varepsilon] = \sigma^2.$$

(仮定 3) 各観測誤差は互いに独立であり，その共分散はゼロである．

$$\mathrm{E}[\varepsilon_i \varepsilon_{i'}] = 0,\quad (i \models i',\ i=1,\lll,m,\ i'=1,\lll,m).$$

(仮定 4) 誤差は以下のような正規確率密度関数に従う．

$$p(z) = (2\pi\sigma^2)^{-1/2} \exp[\ \ z^2/2\sigma^2].$$

ここで，$\sigma\ (>0)$ は標準偏差を表す定数である．

以上の仮定の下で，よく知られた Gauss - Markov の定理により，回帰係数 $\boldsymbol{\beta} = (\beta_0,\ \beta_1,\ldots,\ \beta_n)$ の OLS 推定値は

$$\hat{\boldsymbol{\beta}}_{OLS} = \min_{\boldsymbol{\beta}} \sum_{i=1}^{m} \left(y_i \quad \sum_{j=0}^{n} \beta_j x_{i,j} \right)^2, \tag{2}$$

によって与えられる．ここで，y_i と $x_{i,j}$ はそれぞれ Y_i と $X_{i,j}$ の観測値である．上述の OLS 推定値は最尤推定値であることは良く知られた事実である．しかし実際には，説明変数 $X_{i,j}$ は互いに強い相関を持つことが多く，**(仮定 3)** の $\mathrm{E}[\varepsilon_i \varepsilon_{i'}] = 0$ となることはむしろ稀である．このような多重共線性が存在する場合においては，OLS 推定値が不安定となることが知られており，変数選択を行うことによって目的変数に対して真に効果のある説明変数のみを残す必要がある．本論文では，赤池情報量基準 (AIC) を尺度としたステップワイズ法を用いる．

2.2 Ridge 回帰と Lasso 回帰

多重共線性に対処するための手法として，Horel and Kennard [15] による Ridge 回帰がある．これは観測データに当てはめたモデルの安定化に寄与する方法として知られている．OLS 回帰は，式 (2) のように誤差の総和 (residual sum of squares; RSS) を最小にすることが目的であるのに対して，Ridge 回帰は回帰係数 $\boldsymbol{\beta}$ の二乗和を罰則項（正則化項）として最小化目的関数に加える．一般的に，Ridge 回帰による回帰係数 $\boldsymbol{\beta}$ の Ridge 推定値は

$$\hat{\boldsymbol{\beta}}_{Rid} = \min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{m} \left(y_i \quad \sum_{j=0}^{n} \beta_j x_{i,j} \right)^2 + s \sum_{j=1}^{n} \beta_j^2 \right\}, \tag{3}$$

によって与えられる．ここで，$s\ (\sim 0)$ は Ridge パラメータと呼ばれ, 回帰係数 $\boldsymbol{\beta}$ に課する縮小量をコントロールするパラメータである．明らかに，$s=0$ の場合は $\hat{\boldsymbol{\beta}}_{Rid} = \hat{\boldsymbol{\beta}}_{OLS}$ となる．一方，s が大きい時に，回帰係数 $\boldsymbol{\beta}$ は 0 に近づくように縮小される．一般的に，Ridge パラメータ s の決定には一般化クロスバリデーション (generalized cross-validation; GCV) 法 [16] が用いられる．GCV 尺度を以下のように定義する．

$$GCV(s) = \frac{1}{m}\sum_{i=1}^{m}(\frac{y_i \quad \hat{y}_i(s)}{1 \quad trace(H)/m})^2. \tag{4}$$

ここで, $trace(H)$ は有効パラメータ数, $\hat{y}_i(s)$ は Ridge パラメータ s をある値に設定したときに得られるプロジェクト i の開発労力の推定値を表す．このように，$GCV(s)$ はパラメータ s の関数になっており，最適な s は $\mathrm{argmin}_s GCV(s)$ の解として定義される．Ridge 回帰をソフトウェア開発労力の予測に初めて導入した関連研究は文献 [8] である．

一方，回帰係数 $\boldsymbol{\beta}$ の絶対値の和を罰則項（正則化項）としたのが Tibshirani [17] によって提唱された Lasso (least absolute shrinkage and selection operator) 回帰と呼ばれる推定法である．Ridge 回帰と同様に，Lasso 回帰による回帰係数 $\boldsymbol{\beta}$ の Lasso 推定値は

$$\hat{\boldsymbol{\beta}}_{Las} = \min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{m}\left(y_i \quad \sum_{j=0}^{n}\beta_j x_{i,j}\right)^2 + s\sum_{j=1}^{n}\beta_j\right\}, \tag{5}$$

によって与えられる．Ridge 罰則項 $s\sum_{j=1}^{n}\beta_j^2$ の代わりに Lasso 罰則項 $s\sum_{j=1}^{n}\beta_j$ が用いられていることに注意されたい．Lasso パラメータ s も上記の GCV 法によって決定することが出来る．これまでに，Lasso 回帰をソフトウェア開発労力の予測に用いた従来研究はなく，本論文でその適用可能性について調べることは有意義であると考える．

3. CBR による予測手法

Mukhopadhyay ら [11] は CBR モデルとして，類推的な問題解決手順を組み込んだ Estor を提案しており，専門家，COCOMO，ファンクションポイントに基づいた評価よりも好ましい予測結果が得られたと報告している．しかし，現行プロジェクトと類似する事例を作成する専門家に依存することが Estor の弱点と言える．

Shepperd and Schofield [12] は専門家を必要としない純然たる CBR 方策を導入し，プロジェクト類似性に基づいた手法を複数のソフトウェア開発プロジェクトデータに適用した．彼らの手法では，まず，過去プロジェクトから現行プロジェクトに類似したものを 3 つ選出し，そして，類似プロジェクトの開発労力の代数的平均を現行プロジェクトの開発労力の予測値としている．類似プロジェクトは n 次元空間における類似度を算出することにより特定出来る．本論文では類似性の評価尺度として，文献 [12] で利用されたユークリッド距離 (Euclidean distance; ED) に加えて，余弦類似度 (Cosine similarity; COS) [13] を用いる．類似度を算出する際，プロジェクト特性ごとの単位の違いが予測に影響を与えないようにする必要がある．まず，過去プロジェクトの開発労力 y_i とプロジェクト特性 $x_{i,j}$ を以下のように正規化する．

$$Z(y_i) = \frac{y_i \quad \frac{1}{m}\sum_{i=1}^{m} y_i}{\sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(y_i \quad \frac{1}{m}\sum_{i=1}^{m} y_i)^2}}, \tag{6}$$

$$Z(x_{i,j}) = \frac{x_{i,j} \quad \frac{1}{m}\sum_{i=1}^{m} x_{i,j}}{\sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_{i,j} \quad \frac{1}{m}\sum_{i=1}^{m} x_{i,j})^2}}. \tag{7}$$

そして，現行プロジェクト a と過去プロジェクト p ($1 \geq p \geq m$) のユークリッド距離 $\mathrm{ED}_{a,p}$ と余弦類似度 $\mathrm{COS}_{a,p}$ はそれぞれ

$$\mathrm{ED}_{a,p} = \sqrt{\sum_{j=1}^{n}(Z(x_{a,j}) \quad Z(x_{p,j}))^2} \tag{8}$$

と

$$\mathrm{COS}_{a,p} = \frac{\sum_{j=1}^{n} Z(x_{a,j}) \leq Z(x_{p,j})}{\sqrt{\sum_{j=1}^{n} Z(x_{a,j})^2}\sqrt{\sum_{j=1}^{n} Z(x_{p,j})^2}} \tag{9}$$

のように定義される．

4. プロジェクト類似性に基づいた線形重回帰モデル

従来の LMR モデルでは，過去の全プロジェクトを入力データとして与えるが，モデルの予測精度は観測データに含まれる外れ値に影響されることが多い．そのため，回帰係数 $\boldsymbol{\beta}$ を求める上で，過去の全プロジェクトではなく類似プロジェクトのみを用いることがより合理的であると考えられる．よって，プロジェクト間の類似性に基づいた線形重回帰 (analogy-based linear multivariate regression; anaLMR) モデルを以下の手続きによって構築する．まず，現行プロジェクトと過去プロジェクトの類似度を計算する．次に，類似度の近い順に過去プロジェクトを並べ替え，最も類似した k 個の過去プロジェクトを選出する．そして，k 個の過去プロジェクトを入力データとして LMR モデルに与え，回帰係数 $\boldsymbol{\beta}$ の推定値を求める．最後に，回帰係数 $\boldsymbol{\beta}$ の推定値と現行プロジェクトのプロジェクト特性の観測値を用いて，現行プロジェクト開発労力の予測値を算出する．よって，anaLMR モデルの回帰係数 $\boldsymbol{\beta}$ の OLS 推定値，Ridge 推定値，Lasso 推定値はそれぞれ以下のように与えられる．

$$\hat{\boldsymbol{\beta}}^{ana}_{OLS} = \min_{\boldsymbol{\beta}}\sum_{i=1}^{k}\left(y_i \quad \sum_{j=0}^{n}\beta_j x_{i,j}\right)^2, \tag{10}$$

$$\hat{\boldsymbol{\beta}}^{ana}_{Ridge} = \min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{k}\left(y_i \quad \sum_{j=0}^{n}\beta_j x_{i,j}\right)^2 + s\sum_{j=1}^{n}\beta_j^2\right\}, \tag{11}$$

$$\hat{\boldsymbol{\beta}}^{ana}_{Las} = \min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{k}\left(y_i \quad \sum_{j=0}^{n}\beta_j x_{i,j}\right)^2 + s\sum_{j=1}^{n}\beta_j\right\}. \tag{12}$$

表 1 Desharnais データセット（一部抜粋).

No. Pro	TeamExp	ManagerExp	Length	Transactions	Entities	PointsAdjust	Envergure	PointsAdjust	Language	Effort
1	1	4	12	253	52	305	34	302	1	5152
2	4	4	1	40	60	100	18	83	1	805
3	2	1	9	119	42	161	25	145	2	2569
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
70	4	4	36	886	241	1127	34	1116	1	23940

一般的に，LMR モデルの予測精度を向上させる方法として，データの前処理に対数変換を行うことが知られている [18]．2.1 節で述べたように, LMR モデルは誤差が正規分布に従うことを前提としている．しかし，ソフトウェア開発プロジェクトの観測データは偏った分布に従うことがある．例えば，Boehm [1] や Desharnais [19] において考察されたソフトウェア開発プロジェクトデータの歪度はそれぞれ 4.37 と 1.97 であった．これらのデータに対し，Kitchenham and Mendes [20] は目的変数である開発労力に加え，説明変数であるトランザクション数などプロジェクト規模に関するいくつかのプロジェクト特性に対数変換を適用すると，観測値の分布が正規分布に近づき，予測精度の高い LMR モデルが得られることを示している．本論文では，データの前処理として，目的変数と全ての説明変数に対数変換を施し，式 (1) の LMR モデルに加え，以下の LogLMR モデル

$$\log Y_i = \sum_{j=0}^{n} \beta_j \log X_{i,j} + \epsilon, \tag{13}$$

を考える．ここで，ϵ は式 (1) と同様な正規誤差項である．よって，LogLMR モデルの回帰係数 $\boldsymbol{\beta}$ の OLS 推定値，Ridge 推定値，Lasso 推定値はそれぞれ，式 (2), 式 (3), 式 (5) の y_i と $x_{i,j}$ の代わりに $\log(y_i)$ と $\log(x_{i,j})$ を用いることによって得ることが出来る．同様に，プロジェクト間の類似性に基づいた LogLMR モデル (以下，anaLogLMR モデルと記す) の回帰係数 $\boldsymbol{\beta}$ の OLS 推定値，Ridge 推定値，Lasso 推定値は式 (10) / (12) の y_i と $x_{i,j}$ の代わりに $\log(y_i)$ と $\log(x_{i,j})$ を用いることによって得られる．

5. 数値例

5.1 データセット

提案モデルの性能評価に用いたデータは Boehm データセット [1] と Desharnais データセット [19] である．これらは PROMISE Software Engineering Repository [21] から入手できる．Boehm データセットは 1970 年から 1981 年までの間に金融や工学分野で使用されていたソフトウェアの開発プロジェクトデータを 63 件含んでおり，COCOMO の評価にも用いられている．記録されている 16 個のプロジェクト特性は信頼度 (rely)，データベースのサイズ (data)，複雑度 (cplx)，CPU 時間制約条件 (time)，主記憶装置制約条件 (stor)，仮想記憶装置不安定性 (virt)，改良時間 (turn)，分析者能力 (acap)，応用経験 (aexp)，プログラマー能力 (pcap)，仮想記憶装置経験 (vexp)，プログラミング言語経験 (lexp)，プログラミング熟練度 (modp)，ソフトウェアツールの使用 (tool)，開発日程制約条件 (sced)，コード行数 (loc) である．Desharnais データセットは 1980 年代にカナダのソフトウェア企業で収集された 81 件のプロジェクトデータを含む．うち 4 件はデータの欠損があり，7 件は 0 を含むため対数変換できないので，本論文では，残りの 70 個のプロジェクトを用いて提案モデルの予測精度を評価する．データセットには 9 個のプロジェクト特性が含まれており，それぞれ，チーム経験 (TeamExp in years), マネージャー経験 (ManagerExp in years), 開発期間 (Length in months), トランザクション数 (Transactions), エンティティ数 (Entities), 調整済ファンクションポイント (PointsAdjust), 規模 (Envergure), 未調整ファンクションポイント (PointsNonAdjust)，開発言語 (Language) である．表 1 に Desharnais データセットの一部を抜粋したものを示す．

5.2 実験手順と評価尺度

実験では，leave-one-out クロスバリデーション法を用いる．具体的な手順を以下に示す．

(1) m 個の過去プロジェクトから一つ選び, 目的プロジェクト a $(1 \geq a \geq m)$ とし，残りの $m \quad 1$ 個のプロジェクトをトレーニングプロジェクト p $(1 \geq p \geq m;\ p \models a)$ とする．

(2) 目的プロジェクト a と全てのトレーニングプロジェクト p の類似度を計算する．

(3) 類似度の近い順にトレーニングプロジェクト p を並べ替え，最も類似した k 個のプロジェクトを選出する．

(4) k 個の類似プロジェクトを入力データとして従来モデル又は提案モデルに代入し，目的プロジェクト a の開発労力を予測する．

(5) ステップ (1) / (4) をプロジェクト $a = 1,\ 2, \ldots,\ m$

に対して行い, 全プロジェクトの開発労力の予測値 $\hat{y}_a^{(k)}$ $(a = 1,\ 2, \ldots,\ m)$ を求める.

評価基準として，平均相対誤差 (mean magnitude of relative error; MMRE) と Pred_{25} を用いる [8].

$$\mathrm{MMRE} \ = \ 100/m \leq \mathrm{MRE}_i. \tag{14}$$

$$\mathrm{Pred}_{25} \ = \ 100/m \leq \sum_{i=1}^{m} \begin{cases} 1, \text{ if } \mathrm{MRE}_i \geq 25\%, \\ 0, \text{ otherwise.} \end{cases} \tag{15}$$

ここで,

$$\mathrm{MRE}_i \ = \ y_i \quad \hat{y}_i^{(k)} \ / y_i \tag{16}$$

は相対誤差であり，y_i と $\hat{y}_i^{(k)}$ はそれぞれプロジェクト i の開発労力の実測値と予測値を表す．Pred_{25} は全プロジェクトの中で，MRE の値が 0.25 以下で予測することが出来たプロジェクトの占める割合を表しており，値が大きい程高い精度で予測出来ることを表す.

5.3 類似プロジェクト数 k の決め方

表 2 に Desharnais データセットのプロジェクト間の類似度ユークリッド距離 (ED) を示す．各行は目的プロジェクト a を，各列はトレーニングプロジェクト p を表しており，例えば，$(a = 1,\ p = 2)$ のマスはプロジェクト 1 を目的プロジェクトとした時に，プロジェクト 2 との ED の値を示してる．表から類推出来るように，類似プロジェクト数 k を一律に決めてしまうと，プロジェクト目的ごとに選出されるプロジェクトの類似度の範囲にかなりの違いが生じる．例えば，プロジェクト $a = 1$ が 69 個のトレーニングプロジェクトを用いて計算した ED の値の最小値は 1.573 で，最大値は 8.732 であるのに対して，プロジェクト $a = 70$ の場合は最小値が 5.247 で，最大値が 13.004 である．仮に $k = 20$ とすれば，プロジェクト $a = 1$ の類似プロジェクトの中で最もユークリッド距離が遠いプロジェクトとの類似度は ED = 2.836 であるのに対し, プロジェクト $a = 70$ と最も遠いプロジェクトの類似度は ED = 9.875 になってしまう.

本論文では，類似プロジェクト数 k を一律に設定する代わりに，ED がある値以下なら類似プロジェクトと見なす．表 2 に示した ED を並べ替え，最小値及び第 1 から第 4 四分位数を $Q_0,\ Q_1, \ldots,\ Q_4$ で表すと，$Q_0 = 0.665$, $Q_1 = 2.820$, $Q_2 = 3.681$, $Q_3 = 4.728$, $Q_4 = 13.004$ になる．説明の便宜上，ED が第 l 四分位数以下のプロジェクトの数を $N_{ED}(Q_l)$ で表し，類似プロジェクト数 k を $k \geq N_{ED}(Q_l)$ に設定して推定を行う．一方，余弦類似度 (COS) は，両プロジェクトが完全に一致する場合は COS = 1 で，全く逆の場合は COS = 1 となる．従って，COS がある値以上なら類似プロジェクトと見なす.

表 2 プロジェクト間の類似度 (ED).
(Desharnais データセットの一部抜粋)

	$p = 1$	$p = 2$	$p = 3$	...	$p = 70$
$a = 1$	-	3.957	3.204	...	8.732
$a = 2$	3.979	-	3.437	...	11.075
$a = 3$	3.212	3.429	-	...	10.335
⋮	⋮	⋮	⋮	⋮	⋮
$a = 70$	10.029	12.726	11.856	...	-

表 3 プロジェクト間の類似度の統計量 (ED を用いた場合).

	データセット	Q_0	Q_1	Q_2	Q_3	Q_4
観測	Boehm [1]	0.018	4.499	5.353	6.426	13.949
データ	Desharnais [19]	0.665	2.820	3.681	4.728	13.004
対数変換	Boehm [1]	0.068	4.627	5.429	6.392	10.171
後データ	Desharnais [19]	0.573	3.144	3.951	4.885	10.018

表 4 プロジェクト間の類似度の統計量 (COS を用いた場合).

	データセット	Q_0	Q_1	Q_2	Q_3	Q_4
観測	Boehm [1]	-0.806	-0.259	-0.016	0.248	1.000
データ	Desharnais [19]	-0.960	-0.344	0.005	0.367	0.965
対数変換	Boehm [1]	-0.803	-0.249	-0.019	0.236	1.000
後データ	Desharnais [19]	-0.957	-0.352	-0.007	0.343	0.981

COS が第 l 四分位数以上のプロジェクトの数を $N_{COS}(Q_l)$ で表し，類似プロジェクト数 k を $k \sim N_{COS}(Q_l)$ に設定して推定を行う．参考のために，表 3 と表 4 にそれぞれ ED と COS を用いた場合の各データセットにおける $Q_0,\ Q_1, \ldots,\ Q_4$ をまとめた.

5.4 結果と考察

表 5 と表 6 に提案モデル anaLMR を用いて各データセットに対する予測を行った結果を示す．一番右の列は全プロジェクトデータに anaLMR モデルを適用することを示しており，従来の LMR モデルによる評価結果と一致する．表中各行の最良な評価値を下線で記しており，それぞれのデータセットにおける MMRE と Pred_{25} の最良値はアスタリスク (*) で印している．まず，各行の最良値が一番右の列 (従来の LMR モデル) に位置しないことに注目されたい．特に表 6 のほとんど全ての行で，最良値は類似プロジェクト数 k を最も小さく設定した場合 $(k \sim N_{COS}(Q_3))$ において得られたことが確認できる．これは，類似プロジェクト数 k を少なくすればする程提案モデルの性能が良くなることを意味するわけではなく，従来の LMR モデルのように全プロジェクトを用いるよりも，類似プロジェクトのみを使用した方が予測精度の向上が期待できると考

表 5 ユークリッド距離 (ED) を用いた anaLMR モデルによる評価結果.

データセット	回帰	評価尺度	$k \leq N_{ED}(Q_1)$	$k \leq N_{ED}(Q_2)$	$k \leq N_{ED}(Q_3)$	$k \leq N_{ED}(Q_4)$ (従来の LMR モデル)
Boehm [1]	OLS	MMRE	922.651	964.299	490.544	715.215
		Pred_{25}	0.000	14.706	15.000	8.889
	Ridge	MMRE	449.737	580.326	490.990	628.443
		Pred_{25}	14.815	15.556	18.000	26.000
	Lasso	MMRE	* 381.095	461.192	823.590	596.443
		Pred_{25}	12.500	10.638	* 24.000	16.327
Desharnais [19]	OLS	MMRE	45.315	45.346	53.404	59.924
		Pred_{25}	32.609	40.103	29.851	30.882
	Ridge	MMRE	* 40.845	48.647	50.500	53.024
		Pred_{25}	* 47.917	37.288	38.235	37.143
	Lasso	MMRE	45.071	44.835	49.818	53.023
		Pred_{25}	43.750	40.678	41.791	40.029

表 6 余弦類似度 (COS) を用いた anaLMR モデルによる評価結果.

データセット	回帰	評価尺度	$k \geq N_{COS}(Q_3)$	$k \geq N_{COS}(Q_2)$	$k \geq N_{COS}(Q_1)$	$k \geq N_{COS}(Q_0)$ (従来の LMR モデル)
Boehm [1]	OLS	MMRE	214.426	930.041	843.181	715.215
		Pred_{25}	12.500	10.256	8.333	8.889
	Ridge	MMRE	208.754	614.329	521.325	628.443
		Pred_{25}	* 31.250	16.364	20.000	16.000
	Lasso	MMRE	* 189.840	797.279	597.825	596.443
		Pred_{25}	23.529	14.035	19.231	16.327
Desharnais [19]	OLS	MMRE	57.796	64.218	60.490	59.924
		Pred_{25}	30.909	33.333	36.232	30.882
	Ridge	MMRE	52.461	58.864	60.088	53.024
		Pred_{25}	44.828	37.143	37.143	37.143
	Lasso	MMRE	* 51.493	62.090	56.373	53.023
		Pred_{25}	* 48.214	35.714	41.429	40.029

えられる．次に，列ごとに OLS 回帰，Ridge 回帰，Lasso 回帰を比較してみると，Ridge 回帰又は Lasso 回帰が最も小さい MMRE 又は最も大きい Pred_{25} を示すことが分かる．特に，Lasso 回帰の予測精度は ED や COS の類似度に限らず，OLS 回帰より優れており，Ridge 回帰をやや上回る結果となっている．

次に，観測データの対数変換を伴う提案モデル anaLogLMR による評価結果を表 7 と表 8 に示す．同様に，一番右の列は全プロジェクトデータに anaLogLMR モデルを適用することを示しており，従来の LogLMR モ

表 7 ユークリッド距離 (ED) を用いた anaLogLMR モデルによる評価結果.

データセット	回帰	評価尺度	$k \leq N_{ED}(Q_1)$	$k \leq N_{ED}(Q_2)$	$k \leq N_{ED}(Q_3)$	$k \leq N_{ED}(Q_4)$ (従来の LogLMR モデル)
Boehm [1]	OLS	MMRE	365.078	62.706	60.734	45.803
		Pred_{25}	24.000	46.000	30.645	38.095
	Ridge	MMRE	59.046	51.614	51.810	46.562
		Pred_{25}	36.000	38.000	32.258	38.095
	Lasso	MMRE	64.940	48.335	43.994	* 43.282
		Pred_{25}	44.000	44.000	* 48.387	41.270
Desharnais [19]	OLS	MMRE	57.452	54.916	57.608	52.858
		Pred_{25}	34.615	33.846	37.143	33.286
	Ridge	MMRE	46.695	45.776	51.934	47.098
		Pred_{25}	40.385	* 44.615	38.571	38.571
	Lasso	MMRE	47.744	* 43.506	52.439	46.447
		Pred_{25}	42.308	* 44.615	41.429	41.000

表 8 余弦類似度 (COS) を用いた anaLogLMR モデルによる評価結果.

データセット	回帰	評価尺度	$k \geq N_{COS}(Q_3)$	$k \geq N_{COS}(Q_2)$	$k \geq N_{COS}(Q_1)$	$k \geq N_{COS}(Q_0)$ (従来の LogLMR モデル)
Boehm [1]	OLS	MMRE	107.329	60.441	48.803	45.803
		Pred_{25}	11.111	34.921	39.683	38.095
	Ridge	MMRE	57.831	50.188	46.059	46.562
		Pred_{25}	27.778	28.571	36.508	38.095
	Lasso	MMRE	54.256	* 41.525	43.683	43.282
		Pred_{25}	38.889	39.683	* 46.032	41.270
Desharnais [19]	OLS	MMRE	57.325	51.844	57.583	52.858
		Pred_{25}	33.871	32.857	30.000	33.286
	Ridge	MMRE	49.248	47.854	51.840	47.098
		Pred_{25}	* 41.935	41.429	40.000	38.571
	Lasso	MMRE	47.186	47.415	54.112	* 46.447
		Pred_{25}	* 41.935	41.429	37.143	41.000

デルによる評価結果と一致する．従来研究では，対数変換の有効性に関して理論的根拠は必ずしも明らかではなかったが，経験的に回帰モデルの予測精度の向上につながることが知られていた [20]．Dejaeger [8] らは対数変換を伴う OLS 回帰モデルが 13 種類の既存手法の中で最もよい予測精度を持つことを示した．本研究の実験においても従来の LogLMR モデルに関しては全く同様の結果を確認することが出来た．表 5 と表 7，表 6 と表 8 の一番右の列を見比べてみると，ソフトウェア開発プロジェクトデータの種類や回帰手法の違いに関わらず，対数変換を観測デー

タに適用する方がより小さい MMRE と大きい $Pred_{25}$ を示す．その差は予測誤差 MMRE において顕著となる．特に，Boehm データセットに関しては，生の観測データに直接 OLS 回帰，Ridge 回帰，Lasso 回帰を適用した時，予測誤差 MMRE がかなり大きくなるのに対して，対数変換を適用した場合の予測は比較的安定することを表 7 と表 8 より確認出来る．一方，提案モデル anaLogLMR では対数変換の有効性は一概に言えない結果となっている．表 5 と表 7，表 6 と表 8 のアスタリスクのついている値を見比べれば分かるように，Boehm データセットでは対数変換を適用した場合の MMRE が小さく，$Pred_{25}$ が大きいことが分かる．しかし，Desharnais データセットでは，特にユークリッド距離 ED を用いた提案モデルにおいて対数変換の効果はあまり見られなかった．対数変換を施すのは観測データに含まれる外れ値を除外することが一つの目的であったので，類似度を考慮せず全プロジェクトを用いて解析した従来法ではその効果が大きかったと考えられる．これに対して，提案手法では類似プロジェクトを選出することによって外れ値をある程度除外出来るので，対数変換の有無は予測精度にあまり大きく影響しないと言える．

6. まとめと今後の課題

本論文では，ソフトウェア開発労力を予測するために，類似性を考慮した線形重回帰モデルを提案した．ソフトウェア開発プロジェクトデータを用いた検証実験では，類似性尺度に基づいた提案モデルは従来の OLS 回帰モデル，Ridge 回帰モデル，Lasso 回帰モデルを上回る予測精度を示すことが確認出来た．特に，Lasso 回帰は他の回帰よりも予測誤差が小さく $Pred_{25}$ が大きい結果となっており，ソフトウェア開発労力を予測するための候補としてとても有力であると考えられる．類似度の計算コストは非常に小さいので，労力予測の精度の向上に大きく貢献する可能性が十分にあると結論付けることが出来る．今後は，より多くのデータセットを用いて実験を行うことで，本論文で得られた知見の一般性を検証する予定である．また，ユークリッド距離や余弦類似度以外の類似性評価尺度についても考察し，比較を行う必要があると考えられる．

参考文献

[1] B. W. Boehm, *Software Engineering Economics*, Prentice Hall (1981).

[2] B. W. Boehm, C. Abts, A. W. Brown, *et al.*, *Software Cost Estimation with Cocomo II*, Prentice Hall (2000).

[3] L. H. Putnam, "A general empirical solution to the Macro software sizing and estimation problem," *IEEE Transactions on Software Engineering*, vol. 4, no. 4, pp. 345–361 (1978).

[4] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: a software science validation," *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639–648 (1983).

[5] M. Jorgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37–60 (2004).

[6] S. Chulani, B. Boehm and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 573–583 (1999).

[7] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53 (2007).

[8] K. Dejaeger, W. Verbeke, D. Martens and B. Baesens, "Data mining techniques for software effort estimation: a comparative study," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375–397 (2012).

[9] G. Finnie, G. Witting and J. M. Desharnais, "A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models," *Journal of Systems and Software*, vol. 39, no. 3, pp. 281–289 (1997).

[10] T. V. Gestel, J. Suykens, B. Baesens, *et al.*, "Benchmarking least squares support vector machine classifiers," *Machine Learning*, vol. 54, no. 1, pp. 5–32 (2004).

[11] T. Mukhopadhyay, S. S. Vicinanza and M. J. Prietula, "Examining the feasibility of a case-based reasoning model for software effort estimation," *MIS Quarterly*, vol. 16, no. 2, pp. 155–171 (1992).

[12] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 12, pp. 736–743 (1997).

[13] N. Ohsugi, M. Tsunoda, A. Monden and K. Matsumoto, "Effort estimation based on collaborative filtering," *Proceedings of the 5th International Conference on Product Focused Software Process Improvement* (PROFES'04), pp. 274–286, Springer, Berlin Heidelberg (2004).

[14] 角田雅照，大杉直樹，門田暁人，松本健一，"協調フィルタリングを用いたソフトウェア開発工数予測方法，" 情報処理学会論文誌, vol. 46, no. 5, pp. 1155–1164 (2005).

[15] A. E. Horel and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67 (1970).

[16] P. Craven and G. Wahba, "Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation," *Numerische Mathematik*, vol. 31, no. 4, pp. 377–403 (1979).

[17] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288 (1996).

[18] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer (2001).

[19] J. M. Desharnais, "Analyse Statistique de la productivitie des projets informatique a partie de la technique des point des fonction," *Masters Thesis, University of Montreal* (1989).

[20] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, Article no. 4 (2009).

[21] PROMISE Software Engineering Repository, URL: http://promise.site.uottawa.ca/SERepository/.