

ソーシャルコーディングにおける ユーザの評判と提案の採否の関係について

江見圭祐 乃村能成
岡山大学大学院 自然科学研究科

2017年1月20日

DPS169

ソーシャルコーディングとは

ソーシャルネットワークの考え方をコーディングに適用

- (1) ソフトウェアプロジェクトのソースコードを公開
- (2) 誰でもプロジェクトを複製し、自由に改変可能
- (3) 全てのユーザはプロジェクトに対して提案を作成可能

提案: バグの修正案や新機能の実装案

GitHub: 代表的なソーシャルコーディングサービス

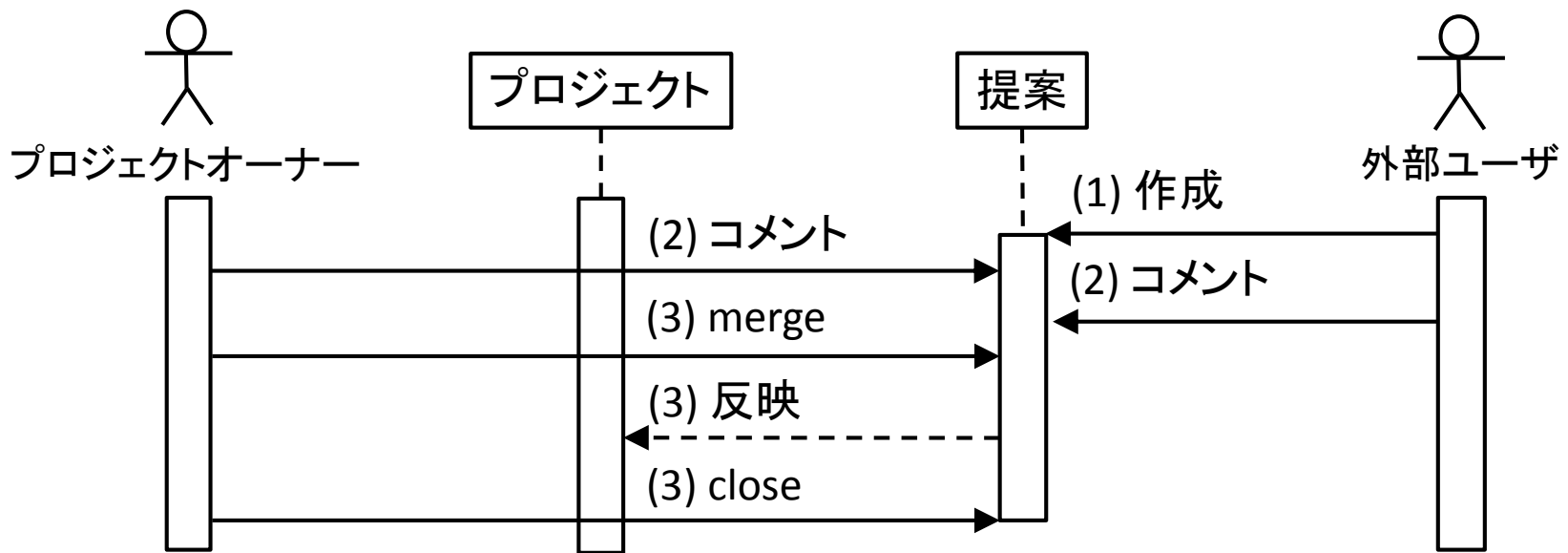
- (1) バージョン管理やコードレビューを支援する機能を提供
- (2) ブランチと呼ばれる機能によって開発の流れを管理



開発の目的ごとにブランチを派生させ、独立に開発

- (3) pull requestによって提案の機能を提供

GitHubにおける提案



<提案の作成以降の流れ>

- (1) 提案を作成し, 変更を反映したいブランチを指定
- (2) プロジェクトに関わるユーザ間で提案について議論
- (3) プロジェクトオーナーが提案の採否を判断

有益提案: プロジェクトに採用(merge)された提案

無益提案: プロジェクトに採用されなかった提案

ソーシャルコーディングでの問題

参加者の多いプロジェクトでは、無益提案が多く作成

➡ 多数の提案の中から有益なものだけを発見する手間大



(既存手法) ユーザの評判や行動を学習し、提案の有益さを算出

ユーザが「誰なのか」に依存する要素

(問題) ユーザの評判が予測結果に強く影響するため、初めてプロジェクトに関わるユーザの作成した提案の予測が困難



ユーザの評判を考慮せずに有益提案を予測する手法を提案

ユーザの評判と提案の採否の関係の調査

プロジェクト名	ユーザ数		提案数		有益提案の割合	
	一見ユーザ	コアユーザ	一見ユーザ	コアユーザ	一見ユーザ	コアユーザ
angular.js	1,747	658	2,405	3,658	40.0%	46.3%
bootstrap	1,772	663	2,435	2,949	30.7%	66.2%
jquery	526	183	709	1,302	25.2%	49.5%
rails	1,825	1,364	3,189	10,408	56.6%	76.1%

一見ユーザ: 初めて提案を作成したユーザ(評判が未知)

コアユーザ: 複数回提案を作成したユーザ(評判が既知)

ユーザ数: 一見ユーザ > コアユーザ

作成した提案数: 一見ユーザ < コアユーザ

有益提案の割合: 一見ユーザ < コアユーザ

参加者の多いプロジェクトの特徴

- (1) 一見ユーザが多く存在し, 無益提案を作成しやすい傾向
- (2) 一部のコアユーザが大量の有益提案を作成

➡ コアユーザの作成する提案ばかりが有益と予測

∴ 一見ユーザの作成する提案は有益と予測されにくい

<プロジェクトオーナーの要求>

ユーザの評判からではなく, 提案の中身で良し悪しを判断したい



ユーザの評判ではなく, ユーザの行動のみから有益提案を予測

提案が拒否される原因の分析

通番	カテゴリ	拒否された理由	個数(のべ)
1	プロジェクトに関する理解不足	不適当なブランチの指定	20個
2		プロジェクトの方針との違い	10個
3		他のPRとの重複	10個
4		別ブランチで解決済み	7個
5		コンフリクト	2個
6	技術的な理解不足	間違った実装	14個
7		テストの不備	5個
8		コーディング規約違反	3個
9	その他	その他	3個

 提案の指定先ブランチが原因で拒否される提案が多い

指定先ブランチが原因で拒否される提案

(1) 不適当なブランチの指定

(例) 安定版のブランチを指定して, 新機能の実装の提案を作成

(2) 別ブランチで解決済み

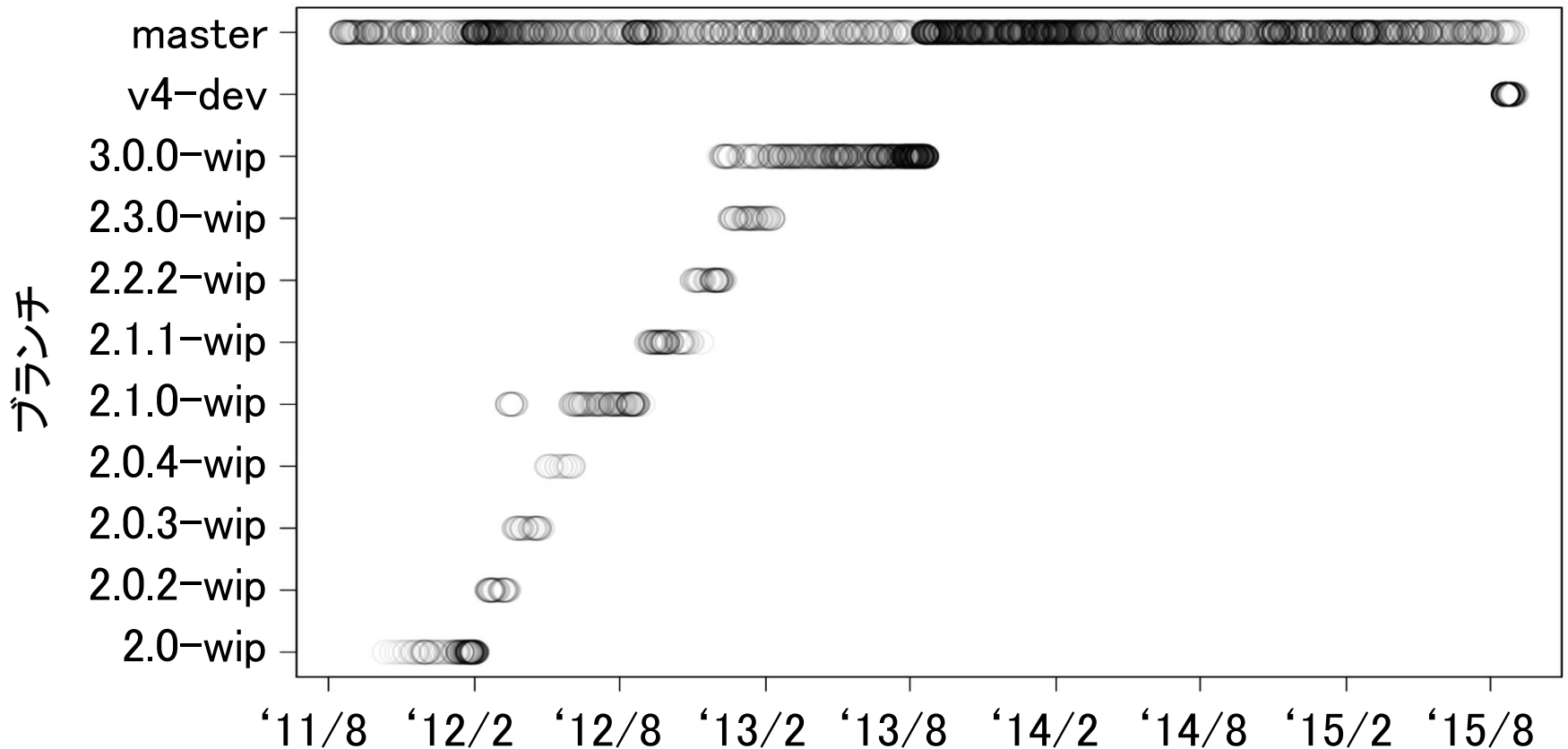
(例) 別ブランチで修正されているバグを修正する提案を作成

 提案の指定先ブランチを学習することで予測できる可能性



提案の指定先ブランチと提案の関係を調査

指定先ブランチと提案の関係

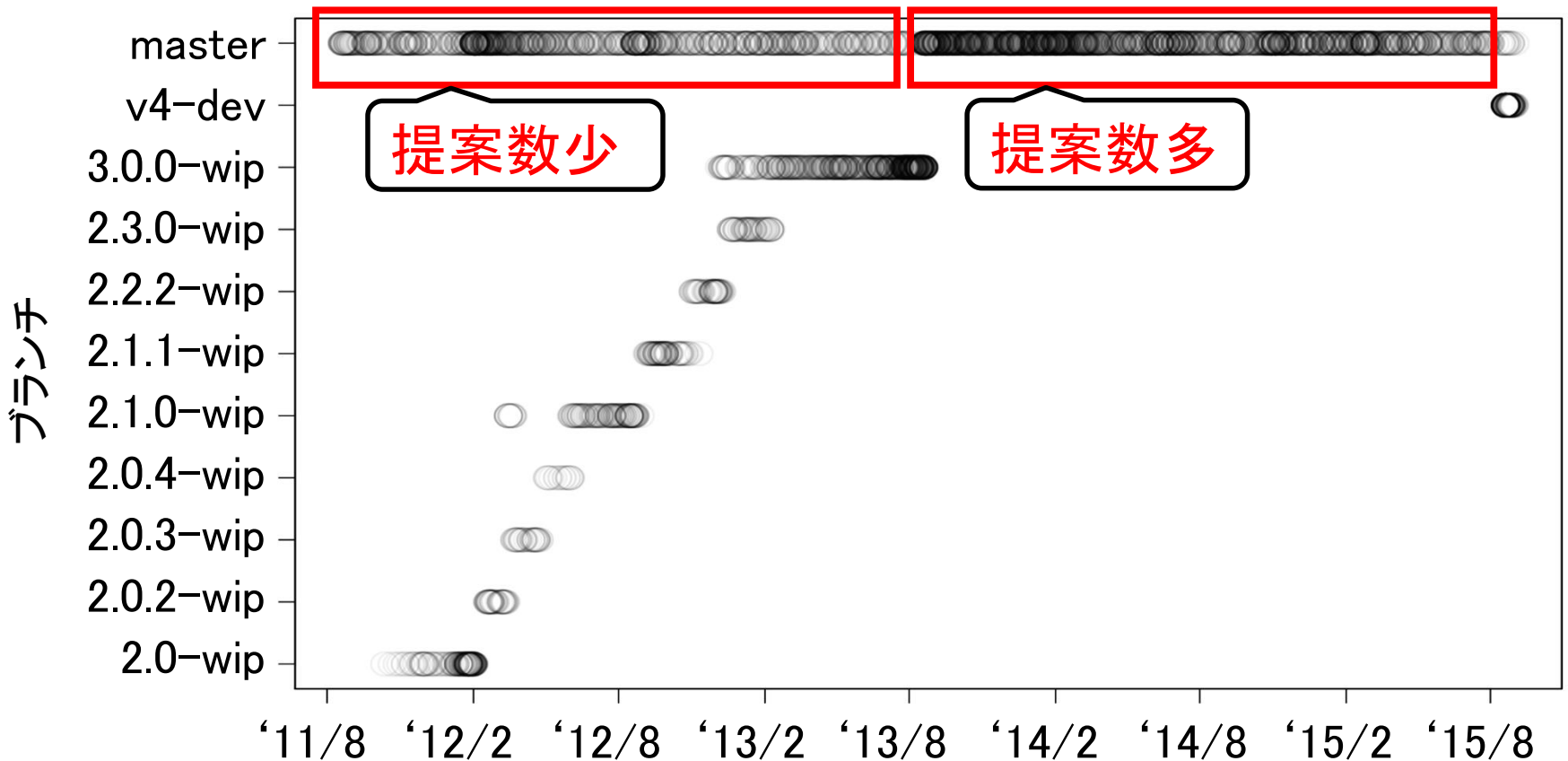


master: リリース用の安定版ブランチ

xxx-wip, xxx-dev: 各バージョンの開発用ブランチ

安定版ブランチだけでなく、開発用ブランチにも多く提案が作成

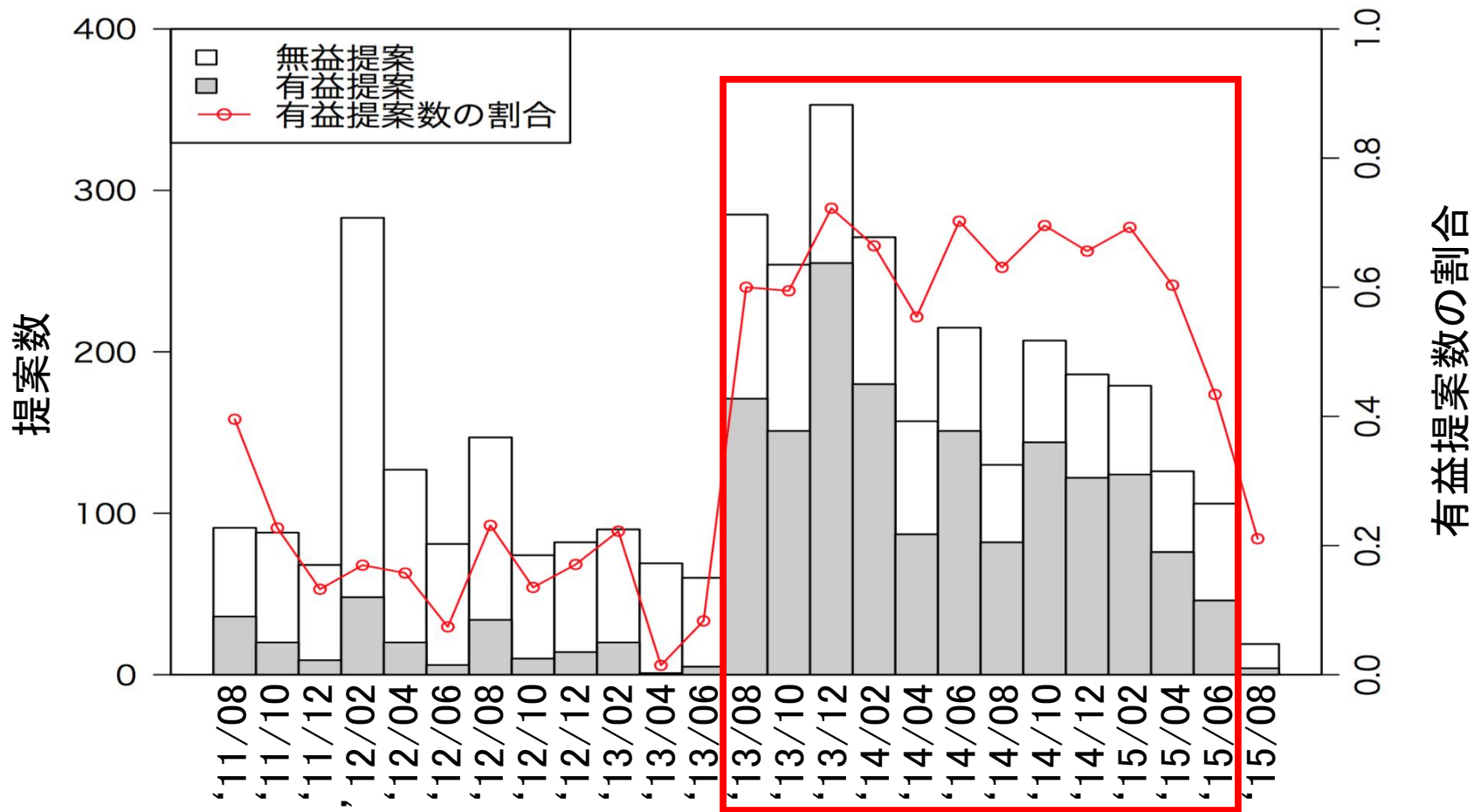
プロジェクトの進捗と提案数の関係



開発用ブランチが存在しない期間は, masterへの提案が急増

➡ プロジェクトの進捗によって提案の傾向が変化

masterブランチに作成された提案数の推移



開発用ブランチが存在しない期間では有益提案数の割合が増加

➡ ブランチがどれほど活発に開発されているかが重要

学習に用いる素性とアルゴリズム

<学習に用いる素性>

- (素性1) 提案が作成されたブランチの過去7日間の変更の回数
- (素性2) 提案で変更したソースコードの行数
- (素性3) 提案で変更したファイルの数
- (素性4) 提案で変更されたファイルの過去3カ月間で変更された回数

<学習に用いるアルゴリズム>

ランダムフォレストを利用

評価

<評価観点>

提案手法を用いることでどの程度有益提案を処理できるか

➡ 提案の有益さを基に提案を整列し、平均適合率を算出

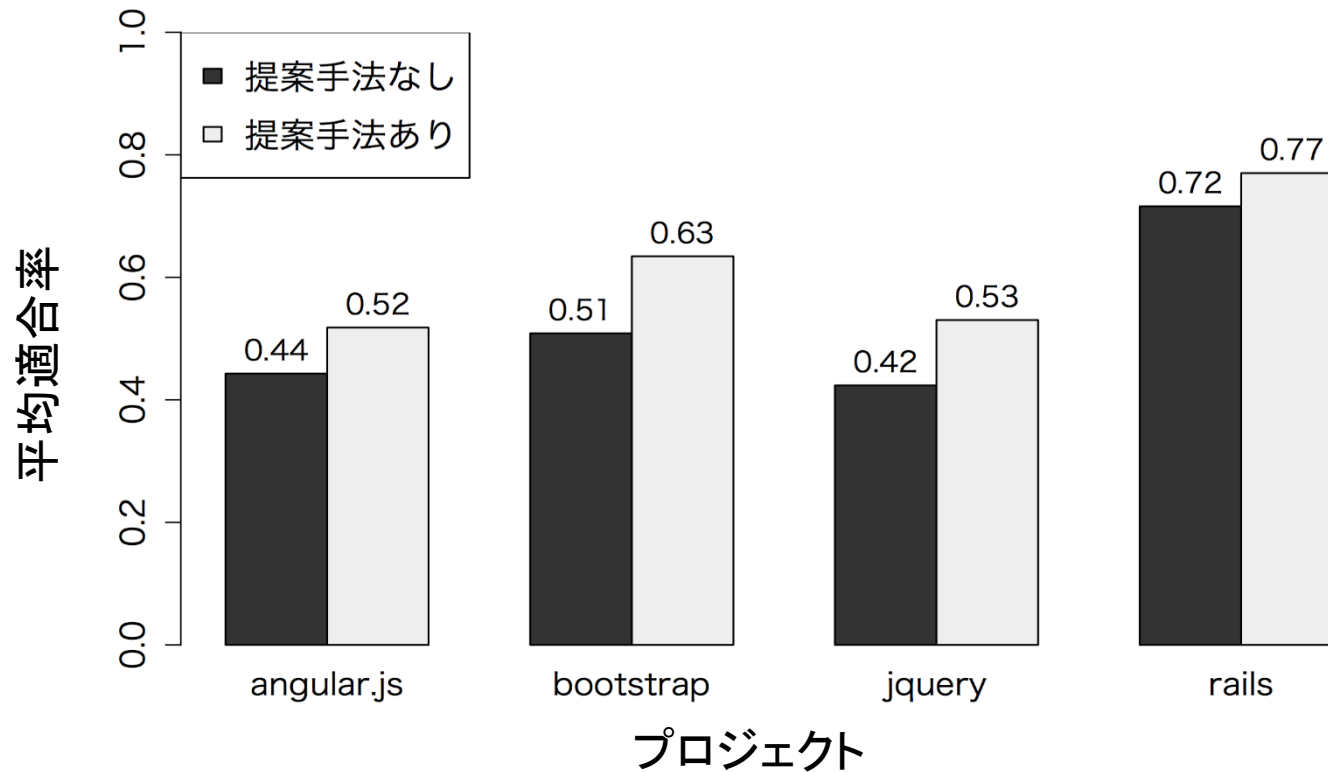
平均適合率: 有益提案が上位に集中しているかを示す指標

<評価環境>

プロジェクト名	総提案数
angular.js	6,963個
bootstrap	5,384個
jquery	2,011個
rails	13,597個

提案の9割を学習データ, 残りをテストデータに分割し, 評価

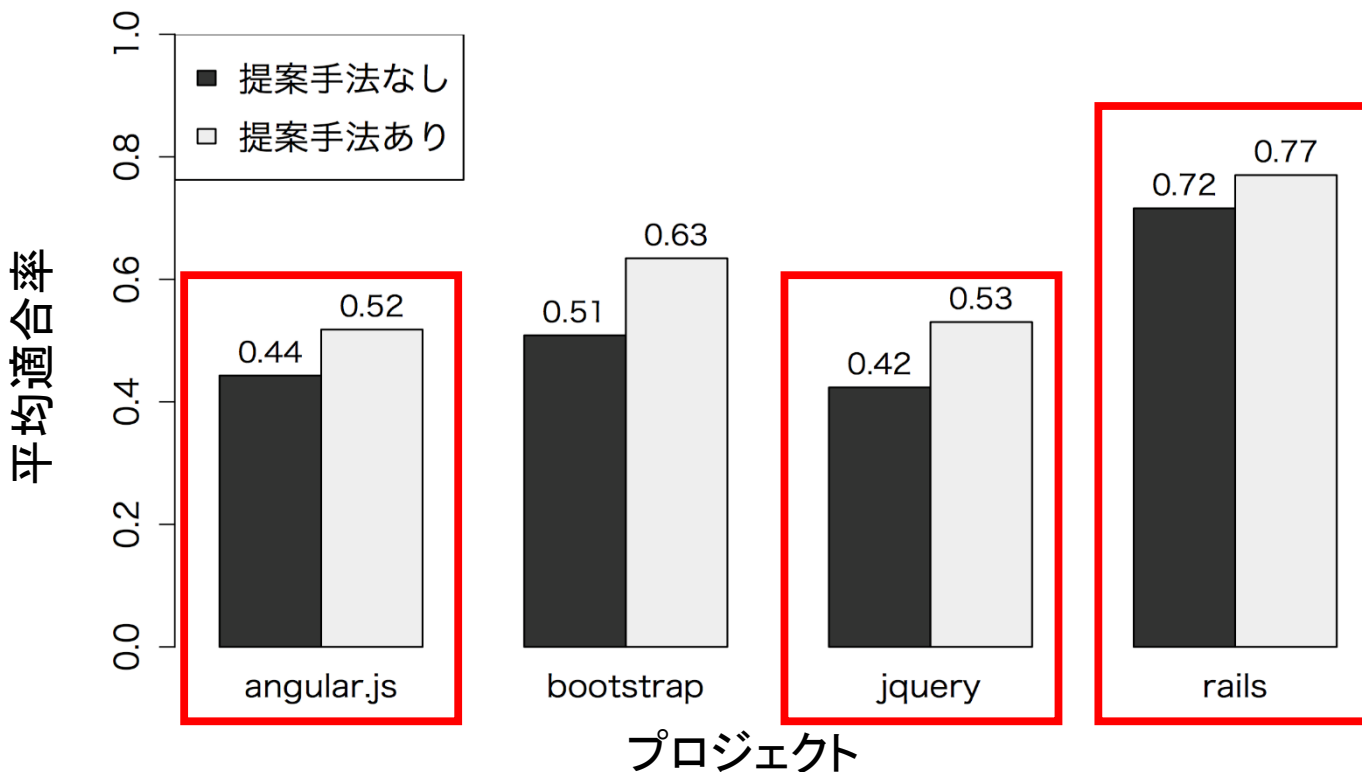
評価結果



すべてのプロジェクトについて、平均適合率が向上

∴ 提案手法を用いることで有益提案を上位に集中可能

プロジェクトの開発体制の違いと予測精度の関係



angular.js, jquery, railsは90%以上の提案がmasterに作成

➡ 指定先ブランチの違いが原因で拒否される提案は少ない

それにも関わらず、すべてのプロジェクトで平均適合率が向上

∴ ブランチの活発さを学習していることが有効に作用 No.15

素性の貢献度

ランダムフォレストを用いて予測に対する素性の貢献度を算出

プロジェクト名	素性1	素性2	素性3	素性4
angular.js	730	431	117	528
bootstrap	723	331	119	533
jquery	218	160	49	173
rails	1,140	641	293	738

すべてのプロジェクトについて(素性1)が最も予測に貢献

(素性1) 提案が作成されたブランチの過去7日間の変更の回数



指定先ブランチが同じであっても、ブランチの活発さによって提案の有益さを予測可能

ユーザの評判に関する考察

提案の採否にはユーザの評判によるバイアスが存在

∴ 提案の採否の判断はプロジェクトオーナーの主観により決定

(例) 貢献の多いユーザの作成した提案を無条件で採用

提案手法で算出された予測値と実際の採否に差異が存在する可能性




(1) 提案手法を用いた予測結果と実際の採否の比較

(2) ユーザの評判がプロジェクトオーナーの判断に及ぼす影響の分析

まとめ

- (1) ユーザの評判と提案の採否の関係を調査
- (2) 提案の指定先ブランチと提案の関係を調査
- (3) ユーザの行動のみから提案の有益さを算出する手法を提案
- (3) 提案手法の評価
 - (A) 提案の順位付けの精度

 ユーザの行動のみから提案の有益さを算出可能