Poster: ACONA: Active Online Model Adaptation for Predicting Continuous Integration Build Failures*

Ansong Ni, Ming Li
National Key Laboratory for Novel Software Technology
Nanjing University
niansong1996@gmail.com,lim@lamda.nju.edu.cn

ABSTRACT

Continuous Integration (CI) reduces risk in software development, but a CI build usually brings huge time and resource consumption. Machine learning methods have been employed to cut the expenses of CI and provide instant feedback by predicting CI results. Nevertheless, effective learning requires massive training data which is not available for a new project. Moreover, due to the diversified characteristics of different projects, reusing models built on other projects leads to poor performance. To address this problem, we propose a novel active online model adaptation approach ACONA, which dynamically adapts a pool of classifiers trained on various projects to a new project using only a small fraction of new data it actively selects. With empirical study on Travis CI, we show that ACONA achieves an improvement of F-Measure by 40.0% while reducing Accumulated Error by 63.2% and the adapted model outperforms existing approaches.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; Risk Management; • **Computing methodologies** → *Learning settings*; *Machine learning approaches*;

KEYWORDS

continuous integration; build failure prediction; online learning; model reuse; active learning

1 INTRODUCTION

In modern software engineering, continuous integration (CI) [1] has become a very popular practice. After developers push their commits or submit a pull request, CI system will actively pull the latest code and perform a build, which consists of the compilation, testing, inspection and sometimes, deployment [1]. This ensures potential defects brought by this change be found immediately thus greatly reduces the risk. However, previous research [4][3] has shown that this process may take several hours or even days to complete for projects at a large scale. Moreover, CI also demands great computation resources for processes like compiling and testing as well as storage space for the VM and dependencies.

Though CI performs an expensive build for every change to identify a potential defective one, the majority of builds tend to be clear [4]. Machine learning has been leveraged to predict CI build outcomes, which provides instant feedback by accurate prediction and reduces the costs of those unnecessary builds [3][5]. Although satisfactory results were achieved by previous works, most of them are based on a well-collected, static dataset which mismatch the reality of CI. In CI practice, build tasks are produced and accumulated along time as data stream, thus existing methods will have to wait for a long time until enough data is collected. Though massive data from numerous previous projects indicates a potential solution, research has shown that the properties of two software projects can be very diverse, thus model reuse between projects can be very difficult [6].

The challenge is : *How to reuse the well-trained classifiers on massive historical data from other projects to help us make good predictions on the current project with streaming data?*

To address this challenge, we propose *ACtive ONline Adaptation* (ACONA). ACONA utilizes a pool of classifiers trained on existing projects. With streaming data from a new project, ACONA tries to select the classifiers that generalize well on the new project and update their weights accordingly. As labels are required to evaluate and select those effective classifiers and they can only be attained by performing real CI builds, ACONA minimizes the demand for labeled data with active learning and selects the most valuable builds to query real CI system for labels to update.

2 METHODOLOGY

ACONA firstly constructs a pool of models trained on a large number of existing projects, and then it elaborates to reuse these well-trained models to a new projects by learning the model adaptation weights using online active learning as the data from the new project keeps on arriving in streams. Such adaptation weights can be regarded as a project-specific "policy" on how to reuse the existing models on this particular project.

2.1 Model Pool Construction and Reuse

With k previous projects and k classifiers trained on them separately, ACONA utilizes a pool of these previously built models. In this work we use Random Forest (RF) as the underlying classifier. As a new project started, ACONA attempt to select those models which generalize well on this new project from the pool. ACONA does this by learning a combination weight $\mathbf{w} \in \mathbb{R}^k$ of all the previously trained classifiers. The combination weights serve as a guideline to reuse the model pool. While adapting to the new project, the

*Research supported by National Key Research and Development Program (2017YFB1001903) and NSFC (61422304).

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195012

models that generalize well are given higher weights and those perform poorly are given almost zero weights.

2.2 Online Adaptation

However, with the special format of streaming data for build tasks, at each time step t, we can only receive one instance $\{\mathbf{x}^{(t)}, y^{(t)}\}$. With offline approaches, models have to be retrained every time the dataset is enlarged by a new instance, which inevitably incurs huge computation cost. To avoid this, ACONA performs the adaptation in an online style using the Generalized Infinitesimal Gradient Descent [7] which is much faster and more effective. The update rule is given in formula 1, with $L(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, y^{(t)})$ denoting the loss function of the SVM.

$$w^{(t+1)} = w^{(t)} - \frac{1}{\sqrt{t}} \cdot \nabla L(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, y^{(t)})_{w^{(t)}} \quad (1)$$

Based on the "no regret theorem" from the work by Zinkevich [7], ACONA is proved to converge to the optimal solution learned by the offline models, but with much faster computation.

2.3 Active Learning

With online optimization, ACONA is guaranteed to converge to an "offline optimal". However, it requires results of every CI builds for update, which can only be obtained by performing actual CI builds. We address this problem by employing active learning in ACONA.

Active learning is widely used in applications where obtaining the labels is costly as in our case. ACONA select the most valuable instances for querying labels by choosing those which are closer to the decision boundary. In other words, builds that are classified with low confidence will be performed by the real CI server to obtain results for update, which is obviously reasonable.

3 EXPERIMENT

With empirical study on *Travis-ci*[*], we attempt to answer the following research questions:

RQ1: *How effective is ACONA in reusing previous models to a new project by adaptation?*

RQ2: *How well does ACONA perform against existing methods?*

3.1 Experiment Settings

We collect the data from *TravisTorrent* which synthesizes the information from *Github* and *Travis-ci*. The dataset we use in this work consists of 534 projects with total amount of 365,766 builds. We pre-process the dataset and extract the features same as our previous work [5] as they are proved to be very effective.

Moreover, we use the following metrics for evaluation, "↑" means the higher the better and "↓" means the lower the better:

- **F-Measure (↑):** It is the harmonic average of *precision* and *recall*. As the class of failed builds is critical in CI outcome prediction, we focus on the F-Measure of it.
- **Accumulated Error (↓):** With an online setting, *Accumulated Error* is a widely used metric for evaluation along the time. It is the average error rate of all t predictions at time t.
- **Query Rate (↓):** Since querying labels from real CI system brings cost, in our experiments, query rate is used to evaluate the percentage of labeled data a model demands for learning.

[*]https://travis-ci.org/

3.2 RQ1: Adaptation Effectiveness

Table 1 shows the improvement on the pool of previously built models by adaptation with ACONA.

Table 1: ACONA Effectiveness

Time Step	F-Measure ↑		Ac. Error ↓	
	Value	Imp. %	Value	Imp. %
Before ACONA	0.310	-	0.492	-
After 50 Builds	0.360	16.1%	0.173	-64.8%
After 100 Builds	0.395	27.4%	0.172	-65.0%
After 200 Builds	0.418	34.8%	0.184	-62.6%
After All Builds	0.434	40.0%	0.181	-63.2%

Answer to RQ1: By learning a selection of classifiers from the pool of previous models with an online optimization over the combination weights of them, ACONA increases *F-Measure* by 40.0% and reduces Accumulated Error by 63.2% for the ensemble of previous models.

3.3 RQ2: Comparison with Existing Methods

Table 2 shows the performance of ACONA against other methods for CI outcome prediction, including *Hoeffding Tree* [2], *Online SVM*, *J48* [3] and *Random Forest*.

Table 2: Performance of Compared Methods

Measurements	ACONA	HT	O-SVM	J48	RF
F-Measure ↑	0.434	0.405	0.227	0.409	0.439
Ac. Error ↓	0.181	0.391	0.268	0.241	0.270
Query Rate ↓	26.5%	100%	100%	100%	100%

Answer to RQ2: Utilizing the model pool helps ACONA learn quicker while active learning policy greatly reduces its demand of labeled data. Thus ACONA achieves better performance with less queries to the real CI system, which indicates that ACONA can cut the expense of CI much better than existing methods.

REFERENCES

[1] Paul M Duvall, Steve Matyas, and Andrew Glover. 2007. *Continuous integration: improving software quality and reducing risk*. Pearson Education.

[2] Jacqui Finlay, Russel Pears, and Andy M Connor. 2014. Data stream mining for predicting software build outcomes using source code metrics. *Information and Software Technology* 56, 2 (2014), 183–198.

[3] Ahmed E Hassan and Ken Zhang. 2006. Using decision trees to predict the certification result of a build. In *21st IEEE/ACM International Conference on Automated Software Engineering, 2006. ASE'06*. IEEE, 189–198.

[4] Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. 2016. Usage, costs, and benefits of continuous integration in open-source projects. In *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on*. IEEE, 426–437.

[5] Ansong Ni and Ming Li. 2017. Cost-effective build outcome prediction using cascaded classifiers. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 455–458.

[6] N. Nagappan T. Zimmermann, E. Giger H. Gall, and B. Murphy. 2009. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 91–100.

[7] Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 928–936.