Sample-based software defect prediction with active and semi-supervised learning

Ming Li · Hongyu Zhang · Rongxin Wu · Zhi-Hua Zhou

Presented by

Ziteng Gao

Overview

Software defect prediction

Groud assumption

The more complex, the more defect-prone

How much complex?

Metrics (What metrics?)

Motivation

Previously prediction methods had fallen into pits,

1. historical or other projects' data dependency
2. assumption on same defect distribution over new projects
3. assumption on same environmental effects exerted on new projects

Sample-based defect prediction with conventional machine learners

Steps as

1. sample a small percentage of modules
2. examine the quality of sampled modules
3. construct a classification model based on the sample
4. predict the unsampled modules in the project

Sample-based defect prediction with conventional machine learners

Conventional learners are

1. Logistic Regression
2. Decision Tree
3. Naive Bayes etc.

Semi-supervised Learning

Learn an initial classifier from a small **labeled training set**

Refine it by further exploiting **unlabeled data**

the CoForest Method

CoForest (Li and Zhou 2007) as

an **ensemble** and **semi-supervised** learner, simply described as

1. Multiple learners (not all same) are trained for the same task
2. If one learner is confident enough on unlable data, it teaches other learners
3. If not confident, nothing happens
4. Repeat steps above until no learners changes

the CoForest algorithm

1. Construct a random forest with N random trees $H = \{h_1, h_2, ..., h_N\}$
2. Repeat 3∼9 until none of the random trees of H changes
3. Update the number of iteration, $t(t = 1, 2, ...)$
4. For each random tree h_i in H, do step 5∼9
5. Construct concomitant ensemble H_{-i}
6. Use H_{-i} to label all the unlabeled data, and estimate the labeling confidence
7. Add the unlabeled data whose labeling confidences are above threshold to a newly labeled set $L^{'}_{t,i}$
8. Undersample $L^{'}_{t,i}$ to make (1) holds. If it does not hold, skip step 9
9. Update h_i by learning a random tree using $L \cup L^{'}_{t,i}$

$$\frac{\hat{e}_{i,t}}{\hat{e}_{i,t-1}} < \frac{W_{i,t-1}}{W_{i,t}} < 1 \tag{1}$$

Flaws of the CoForest Method

Redundant information that learners has already captured,

that is, learners may not learn **new** things from unlabeled data.

Active learning

Active learners can **query label** of unlabeled data **from oracle** (usually domain experts).

Commonly labels that active learners query are **the most useful ones**, and hence the labels they need are usually **less** than normal learners.

the ACoForest Method

What policy?

By **disagreement.**

ACoForest Method takes the advantage of disagreement based **active** learning and traditional **semi-supervised** learning.

the ACoForest Method

...

4. Find M unlabeled examples in U , on whose labeled the random trees in H disagree the most.

5. Query the labels of the selected M unlabeled examples, and places them along with the labels into L.

...

Metrics

Including complexity metrics, abstract syntax trees,

object-oriented metrics, program dependency metrics, etc.

Such as LOC, cyclomatic complexity, number of classes, number of blocks, number of if statements, method references

Evaluation of the performance

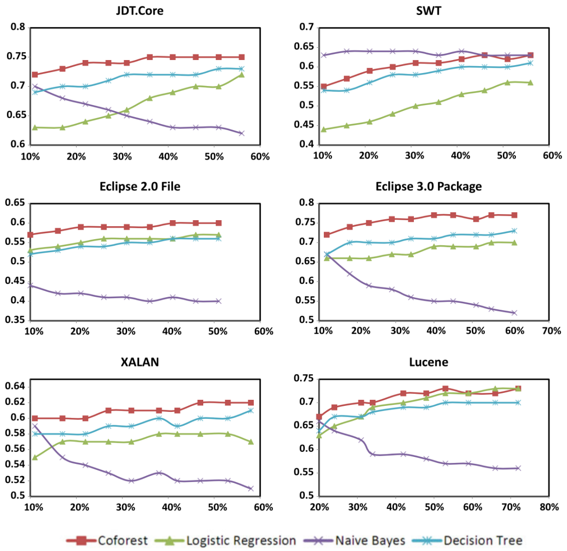
$$P = \frac{tp}{tp + fp}$$
$$R = \frac{tp}{tp + fn}$$
$$F = \frac{2PR}{P + R}$$

P: the Precision

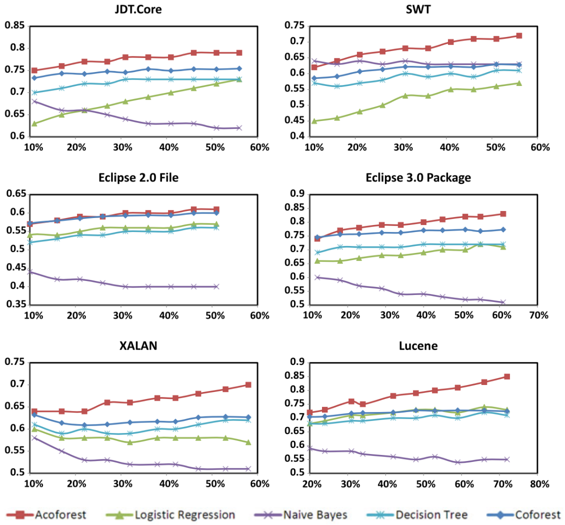
R: the Recall

F: the harmonic mean of P and R

Results of CoForest with others



Results of ACoForest with others


JDT.Core
SWT
Eclipse 2.0 File
Eclipse 3.0 Package
XALAN
Lucene
Acoforest
Logistic Regression
Naive Bayes
Decision Tree
Coforest