Learn to Predict “Affecting Changes” in Software Engineering

Xiaoxia Ren
Dec. 8, 2003
Background

- Software systems evolve over time
  - To adapt to environment
  - To add desired functionality
- Graceful software evolution requires
  - Only expected functionality changes occur
- How to guarantee the correctness?
  - Unit or regression tests
Motivation

- **Given:**
  - Two versions of a program, before and after editing
  - A unit test compilable in both versions

- **Question:**
  - For a specified edit (change), could we predict whether it affects the behavior of this unit test or not?
Data Format

@relation DifferTester.testEmptyPpt1.affectingChanges
@attribute category { ... }
@attribute packageName { ... }
@attribute className { ... }
@attribute signature { ... }
@attribute isAffecting { yes, no }
@data
changeMethod, daikon, PptSlice, check_modbits(), no
addClass, daikon, Dataflow, $1, no
addClass, daikon, diff, none, DepthFirstVisitor, yes
addField, daikon, diff, Diff, PPT_COMPARATOR(java.util.Comparator), yes
deleteField, daikon, PptSlice, parent(daikon, Ppt), no
Experimental Setup

- Data collected from Daikon System over year 2002
- Each week labeled as a version
- Dataset was split at month boundary and each dataset included at least 800 instances
- Prediction based on cumulative data
Accuracy Comparison

![Accuracy Comparison Diagram]
True Positive Rate for “yes” class
Precision for “yes” class
Data Analysis and Some Thoughts

- Why TP and Precision rate so low?
  - Change activity on Daikon was quite diverse. The distribution of training data is different from test data.
  - The percentage of “yes” instance is too small (5.9%)

- Some Thoughts
  - How about a project in a stable stage?
  - Try other ML algorithms?
Q & A?