Discovering Software Bugs with Bayesian Non-Parametric Models

Finale Doshi and Jurgen Van Gael, University of Cambridge

Abstract
We are interested in the problem of identifying bugs in computer programs and additionally patterns of use that cause computer programs to crash. Specifically, we begin with code that has been instrumented to count how many times certain lines are executed during a run through the program. Given successful and failed runs of the program, we are able to identify potential causes of failure (debugged in the Cooperative Bug Isolation project [1]). We assume that the number of times a line is executed in the source code is a good indicator of which lines are more likely to be associated with program failures.

Motivation

Question: Are there particular usage patterns that cause software to fail? Knowing what type of use causes a program to crash may provide useful debugging information.

Approach: Suppose we instrument a program to keep track of various features, for example, how often a certain line of code is executed during a run of the program.

Goal: Find patterns that correspond to usage patterns (such as a mouse click) and determine what usage patterns are associated with program failures.

Instrumenting the Software

The data was obtained from the Cooperative Bug Isolation project [1].

Concept:
Code is instrumented to count the number of times certain lines are executed.

1: print “starting program”
2: filename = get_keyboard_input()
3: action = get_keyboard_input()
4: if action == “load”:
5: read(filename)
6: if action == “save”:
7: write(filename, “some data”) 
8: print “finished execution”

Different usage patterns produce different histograms of counts:

If we consider multiple runs of the program, we will have a histogram containing a mixture of different patterns:

For storage reasons, the counts are subsampled.

Data consists of a few hundred runs with tens of probes; but we can imagine obtaining very large data sets (if many people run instrumented software on their desktops).

Non-Parametric Approach I: DeltaHDP

The hierarchical Dirichlet process is a mixture model of the LDA model, using a Dirichlet process on the number of parts of patterns rather than assuming a fixed number.

Non-Parametric Approach II: IBP

Both the LDA and HDP approach focus on the proportions of probes counts, that is, the shape of the usage pattern histogram, rather than the actual values of the counts.

The Indian Buffet Process places a prior over a space of infinite matrices:

Patterns found by the IBP model:

A larger toy example:
The actual patterns:

Results on Toy Data

More efficient inference procedures exist for it.

Results on Real Data

Could not run IBP model because inference took too long.

Future Work

More efficient inference procedure for the IBP.
For both models, can we handle the data sequentially?
How useful is this for finding bugs in software, and what are other applications?

References: