Review 25

4/24/12

CS6V81.502

Seungtack Baek

**Effective Software Fault Localization**

**Using an RBF Neural Network**

In this Paper, Wong et al. utilizes one of the machine learning techniques, neural network for software fault localization. Throughout this paper, this new technique is constantly compared to other techniques, such as Trantula and other crosstab-based techniques.

One of the main challenges for the neural network is that it needs plethora of data. Yet, the authors of this paper increased the data even more by increasing granularity of the sample from one test case to one statement in the software.

To cut to the point, this neural network approach is shown to be more effective (higher EXAM score) than crosstab-based technique. Also, in evaluation of the technique, both papers evaluated similar subjects, such as GREP and Unix-suite, which makes their comparison more proper.

Even though I could not get into the hard statistic math part, this neural technique seems very noble (and proper even) in that this technique mimics human brain. When I try to spot the faulty part in my application, I do almost same as this technique. I look at the sequence of statement execution that is covered by the test case that ran successfully and compare that with the sequence of statement execution that is covered by faulty test case. Just as this technique, I look at how “faulty” each statement is from test run and try to localize the software fault.

**Question**

1. For each subject, how do they know where bug existed? Did they “injected” a bug and tried to find it?
2. I don’t understand the table, where it says “more effective,” “less effective.” What are these mean?