Software Testing as A Problem of Machine Learning:

Towards a Foundation on Computational Learning Theory

(Extended Abstract of Keynote Speech)

Hong Zhu
School of Engineering, Computing and Mathematics
Oxford Brookes University, Oxford OX33 1HX, UK. Tel. ++44 01865 484580
hzhu@brookes.ac.uk

In recent years, the application of machine learning techniques to software testing has been an active research area. Among the most notable work reported in the literature are those experiments on the uses of supervised and semi-supervised learning techniques to develop test oracles so that the correctness of software outputs and behaviours on new test cases can be predicated [1]. Experiment data show that it seems a promising approach to the test oracle automation problem. In general, software testing is an inductive inference in the course of which the tester attempts to deduce general properties of a software system by observing the behaviours of the system on a finite number of test cases [2]. Thus, there is a great potential for the application of machine learning to software testing.

Since 1980s, researchers have studied the relationships between software testing and inductive inferences. In this talk, I will brief review the main results in this area from a theoretical perspective. The existing work can be roughly classified into two categories. The first is to define test adequacy criteria based on inductive inference techniques. For example, Weyuker proposed an adequacy criterion explicitly involving inductive inference [4]. In particular, a test is defined to be adequate if the program under test can be derived from test cases. More recently, Fraser and Walkingshaw further developed Weyuker's work by employing Probably Approximately Correct inductive inference protocol to define behavioral adequacy criterion, which requires an accurate model of the software to be derivable from adequate test cases [5].

In general, an inductive inference device M is a function. It takes a finite subset X of input/output pairs of a function f on a domain D, as input and produces a function $M(f)$ such that it is correct on the set X of input/output pairs. An adequacy criteria $C_M(t, p)$ can then be defined as $C_M(t,p) \Leftrightarrow M(p \downarrow t) = p$, where t is a finite test set, p is a function on D under test, $p \downarrow t$ is the subset of input/output pairs of p with input from t. Employing the identification in the limit protocol, the following were proved [3].

Theorem 1. A program p is correct *w.r.t.* specification s after successfully tested on a finite test set t, if t is adequate according to criterion $C_M(t, p)$, p is explanatorily learnable by M, s is behaviourally learnable by M, and M converges to a function that is consistent with s on t. □

Theorem 2. If both program p and specification s belong to a set of functions that are learnable by identification in the limit, the correctness of program p w.r.t. specification s can be determined by testing on a finite number of test cases. □

The main conclusions that we can draw are two folds. First, a function is learnable implies that it is testable. Thus, learning is a more difficult computational problem than testing. Second, when a machine learning technique is used for test automation, its inductive inference power (i.e. the set of functions that is learnable for the inference device) determines the set of functions that are testable.

Similarly, given a Probably Approximately Correct (PAC) inference machine M, we can define an test adequacy measurement $K_{M,\delta}(t, p)$, which is a function from test sets t and programs p to real number adequacy scores in the range [0,1]. The following theorem links test adequacy to software reliability.

Theorem 3. For a finite random test set t, the program p is correct on t w.r.t a specification s which is in a set P of functions PAC learnable by M, then the δ-probable reliability of p is $K_{M,\delta}(t, p)$. □

A practical implication of Theorem 3 is that the complexity of the software under test should be taken into consideration in reliability estimation since the complexity of the function determines learnability, and thus testability.

The second category is to analyze existing software testing techniques and methods from an inductive inference point of view. Considering testing as a process of inductive inference, the question is whether the induction converges to a right conclusion when testing stops. Because test adequacy criteria are used as stop rules, the analysis of testing methods can be performed via examining test adequacy criteria using various inductive inference protocols. In [3], Weyuker's axioms of test adequacy criteria were studied with identification in the limit. It was proved that the adequacy criterion $C_M(t, p)$ satisfies Weyuker's axioms if the inference machine M satisfies certain properties, such as conservative. Zhu and Hall's axioms of test adequacy measurement were also examined, but using Valiant's PAC inference protocol. It was proved that the adequacy criterion $K_{M,\delta}(t, p)$ satisfies Zhu and Hall's axioms. In other words, the axiom systems of test adequacy do catch the key properties of inductive inference nature underlying software testing techniques.

REFERENCES

[1] Almaghairbe, R. 2017. *Formulating Test Oracles via Anomaly Detection Techniques*. PhD Thesis, University of Strathclyde, UK.

[2] Zhu, H., Hall, P., and May, J. 1992. Inductive inference and software testing. *Journal of Software Testing, Verification, and Reliability*, 2:69-81.

[3] Zhu, H. 1996. A formal interpretation of software testing as inductive inference. *Journal of Software Testing, Verification and Reliability*, 6:3-31.

[4] Weyuker, E. J. 1983. Assessing test data adequacy through program inference. *ACM Transactions on Programming Languages and Systems*, 5(4), 641-655.

[5] Fraser, G and Neil Walkinshaw, N. 2015. Assessing and generating test sets in terms of behavioural adequacy. *Software Testing, Verification And Reliability,* 25:749–780.