September 15, 2017

Neural network based regression test selection

Komateswar Dhanadevan
Hewlett Packard Enterprise

Jayasankar Nallasamy
Hewlett Packard Enterprise

Sundaravadivel Murugavel
Hewlett Packard Enterprise

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation
Dhanadevan, Komateswar; Nallasamy, Jayasankar; and Murugavel, Sundaravadivel, "Neural network based regression test selection", Technical Disclosure Commons, (September 15, 2017)
http://www.tdcommons.org/dpubs_series/669

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
Neural network based regression test selection

Abstract:

Need for regression testing has been ever growing demand for software development organizations so does the cost of executing regression test suites. The intent of this idea is to propose a unique method in regression test for selecting a subset of test cases as oppose to select all test cases for a given software change using Neural Network. In general, a software consist of multiple features and it interacts with each other in the order of different magnitude. Each feature interaction is assessed based on five relative dimensional complexity. These complexities for each feature interaction is determined and subjected to neural network based training model with appropriate test set. Subsequently any code change in a software iterative build can be evaluated for regression test by providing a new set of change dimensional complexity parameters for the affected features to the machine learning software for predicting optimal number of regression test cases. This method will help the organization to reduce the overall regression test effort significantly.

This disclosure relates to the field of computer software testing and Neural Network based Machine learning.

To overcome regression testing challenges, intelligent testing is a typical solution by way of doing it faster but in reduced time. The focus of this idea is to propose a machine learning based approach to select optimal set of test cases for regression test based on the complexity of software module that has undergone changes.

A software product consist of many features. Software features do not exist in isolation and they interact with each other. The degree of interaction between the features determines the intensity of the friction among them. Any change in the module of that particular feature will proportionately affect the interacting features.

As a first step, developing a software feature interaction matrix and creating a corresponding test coverage map in order to test the interplay between features is of paramount importance. These interaction matrix can be created either at the module levels for each software feature or at the major subsystem level and/or with its eco-system. Each intersection in the Feature interaction matrix represents the complexity of interacting modules in five different dimensions, namely Data, Logic, Environment, Structure and Use case. Here the complexity refers to system that exhibits numerous behaviors.

Data Complexity: The interacting feature module set to have data complexity when its behavior changes based on the input value and therefore generates different results.

Logic or conditional Complexity: The interacting feature module set to have logical complexity when it has quite a few conditions and many paths needs to be tested. Structural Complexity: The interacting feature module set to have structural complexity depends on the distance of interconnection among them (ie. Tightly or loosely coupled) and number of elements that makes
up the module/system. The module/system is set to have structural complexity when they are tightly couple and has more number of dynamic interconnections

Environment Complexity: The interacting feature module set to have environment complexity when the behavior of the module/system depends on the configuration and its operation and rules of the environment.

Use case complexity: The interacting feature module set to have use case complexity when a change affects the behavior of software from different end user perspective.

Using the above complexity analysis, a feed forward neural network as shown in the below Figure, can be built with five dimensional complexity as input and each test case number as an output with binary value (‘0’ or ‘1’). Each complexity dimension can be ranked from 1 to 5 (or as per user defined). 1 being least affected and 5 being heavily dependent. Based on the ranking, prepare a sample dataset of various possible input and output combinations for each feature interaction.

The learning sample dataset is applied to neural network based supervised learning till it reaches the desired target error.

Below Figure depicts the output of an open neural network software that generated a feed forward network with all positive and negative weights. This sample neural network consists of single hidden layer with 3 neurons.

After subjecting the neural network with samples for several cycles of learning, a query has been made with all dimensional complexity set to ‘0’ and neural network successfully have not selected any test cases from that interaction test set. A query to neural network with complexity set to ‘5’ resulted in successful selection of all test cases. Further a query with new combination of value that neural network never seen, resulted in selection of optimum number of test cases for a new set of dimensional complexity.

Advantages:

- Ability to identify accurate and optimal sub-set of regression tests for the corresponding bug-fix/code changes.
- Minimizes cost in terms of effort and time, and enhances the test effectiveness.
- The idea can be leveraged to any software under test and it's easy to build and construct
Disclosed by Komateswar Dhanadevan, Jayasankar Nallasamy, Sundaravadivel Murugavel, Hewlett Packard Enterprise