TESTING MACHINE LEARNING ALGORITHMS WITHOUT ORACLE

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfilment
of the Requirements for the Degree

Master of Science in Computer Science

University of Nebraska at Omaha

by

Abhishek Kumar

August, 2018

Supervisory Committee:

Harvey Siy, Ph.D.

Myoungkyu Song, Ph.D.

Matthew Hale, Ph.D.

TESTING MACHINE LEARNING ALGORITHMS WITHOUT ORACLE

Abhishek Kumar, M. S.

University of Nebraska, 2018

Advisor: Harvey Siy, Ph.D.

Abstract here

ACKNOWLEDGMENTS

Acknowledgments here

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning algorithms are becoming increasingly complex and difficult to implement and analyze. While developing and training an implementation of a machine learning model the aim is to create a model that makes the best predictions. Lack of reliable oracles for testing machine learning algorithms makes it very hard to test such implementations. Such set of programs which does not have a test oracle that can predict the output on a set of inputs are called "non-testable programs"[2].

Chapter 2

Literature Review

2.1 Testing Without Oracles

On testing non-testable programs.

The current testing research activities fall under three categories: Developing a sound theoretical basis for testing. Devising and improving testing methodologies, especially the mechanizable ones. Defining accurate measurement criteria for testing data adequately. An oracle is a system that determines the correctness of the solution by comparing the systems output to the one that it predicts. A program is considered non-testable if one of the following two conditions occur: A oracle does not exist for the given problem. It is theoretically possible to determine the correct output but computationally very hard. The programs that dont have oracles can be usually classified in three categories: Programs that were written to determine the correct answer. Programs that produce lot of outputs such that it is hard to verify all of them. Programs where tester have a misconceptions (tester believes that he has the oracle even though he might not). Pseudo Oracles/Dual Coding: Another set of program is written independently according to the same specification as the original program and the output from both the programs is compared. If the outputs match it can be asserted that the original results are according to the specification. The problem with dual coding is that it has a lot of overhead and requires more time and money. This is done only for highly critical softwares. While performing mathematical computations, errors from three sources can creep in: The mathematical model used to do the computations. Programs written to implement the computation. The features of the environment like: round-off, floating point operations etc. Even in the absence of oracles the users often have a ballpark idea of what the correct answer would look like without knowing the correct answer. In such cases we make use of partial oracles. It is relatively easier to test the systems on simpler inputs for which the output is known. The problem, of course, is that from experience we know that most errors occur in complicated test-cases. It is common for central test cases to work and boundary cases to fail. From the above observations the authors make five recommendation for items to be considered as a part of documentation. The criteria used to select the test data. The degree to which the criteria was fulfilled. The test data, the program ran on. The output of each of each test datum. How the results were determined to be correct or acceptable. Although the recommendations do not solve the problem of non-testable programs but they do provide information on whether the program should be considered adequately tested or not.

2.2 Metamorphic Testing

2.3 Overview of Machine Learning Algorithms

2.4 Testing Machine Learning Programs

Chapter 3

Proposed Work

3.1 Setting up the Test Environment

3.1.1 Docker

For replication of results. Image can be downloaded from dockerhub. Attached volume for persisting data.

3.1.2 Jupyter

3.1.3 Tensorflow

3.2 Selection of Implementations to Test

Chapter 4

Work Plan

Bibliography

[1] T. Y. Chen, F. C. Kuo, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing and beyond. In *Proceedings - 11th Annual International Workshop on Software Technology and Engineering Practice, STEP 2003*, pages 94–100, 2004.

[2] Christian Murphy, Gail Kaiser, and Lifeng Hu. Properties of Machine Learning Applications for Use in Metamorphic Testing.

[3] S Nakajima and H N Bui. Dataset Coverage for Testing Machine Learning Computer Programs. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, pages 297–304, 2016.

[4] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortes. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016.

[5] Elaine J. Weyuker. On testing non-testable programs. *Computer Journal*, 25(4):465–470, 1982.

[6] Xiaoyuan Xie, Joshua Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. Application of metamorphic testing to supervised classifiers. In *Proceedings - International Conference on Quality Software*, 2009.

[7] Xiaoyuan Xie, Joshua W K Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. Testing and Validating Machine Learning Classifiers by Metamorphic Testing. *J. Syst. Softw.*, 84(4):544–558, apr 2011.