Advanced Topics in Software Engineering - Handin 4

Jack Neilson

November 23, 2018

Contents

1 Introduction

1.1 Problem Description

Predicting the cost of a project is an important problem to solve, as it allows for businesses and other entities to accurately prioritise and assess the feasibility of potential projects they are considering. There are several ways to predict the cost of a problem - heuristics, expert knowledge, statistical analysis, and so on. This paper will examine the use of genetic programming with respect to predicting the cost of a project given several data points about the project.

1.2 Genetic Programming

Genetic programming is a method of generating programs which can accurately solve a given problem. In this case, the aim is to generate a program which, given several inputs about a project, will accurately predict the cost of said project. To achieve this, a tree structure will be generated using a set of primitive operators which will be applied to the endpoints of the tree. The endpoints of the tree are the inputs given to the program. This tree is then compiled to a runnable function and given the data points from the project. The function will return the estimation for the cost of the project. These trees are then cross-bred with other trees from the population of potential solutions, mutated, and then evaluated again. After a given number of iterations, the best tree is taken as the solution.

1.3 Data Sets

The data sets used to evaluate the genetic programming algorithm can be found in the "data" folder. They were selected because they contained only numeric data, allowing for much easier predictions using genetic programming. Some of the data sets have been modified to exclude the project ID field, as the ID of the project has no bearing on its cost.

2 Implementation

2.1 deap

To implement the genetic programming algorithm, the "deap" library for python was used. It provides several helpful tools, and allows a user to easily register primitive operators for use when generating trees, generate those trees, evaluate those trees, and keep a record of the best trees generated.

2.2 Splitting the Data Set

To prevent the problem of overfitting, where the model generated from a set of data fits that set perfectly but does not accurately capture the true underlying features, the data must be split in to both training and testing sets. The model is trained on the training set, and evaluated using the testing set. To accomplish this, k-folds validation was used. The data split is split into k sets, with one set being taken as the testing set and every other set used to train the model. In this example, the data is split into 10 sets, and the genetic programming algorithm is ran 10 times with each training/testing set. The end result of the algorithm is the mean of those 10 runs. By using this approach the problem of potentially biased training/testing sets is mitigated.

2.3 Evaluation Methods

Two evaluation methods were investigated in this implementation, mean absolute error (MAE) and root mean squared error (RMSE).

The formula for MAE takes the form of:

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - x_i \mid}{n}$$

This is a somewhat naive way of measuring the error, as larger errors are not given as much importance as smaller errors.

Perhaps more appropriate for the problem at hand is RSME, which takes the form of:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}}$$

The root mean squared error gives more importance to larger errors, which is more important in this specific domain as a slight, consistent increase to error is preferable

to a solution which gives a slightly more accurate measure of cost for most data points, but has estimations with a much higher error.

2.4 Linear Regression

As a comparison to the genetic programming algorithm, a linear regression model was created alongside. In the linear regression model, each input variable is given a coefficient depending on how related that input variable is with the cost. These coefficients can then be used to predict the cost when given future input variables (in this case, the test data points).

3 Results

3.1 Mean Absolute Error

3.1.1 Albrecht

3.1.2 China

3.1.3 Kemerer

3.1.4 Miyazaki (94)

3.2 Root Mean Squared Error

3.2.1 Albrecht

3.2.2 China

3.2.3 Kemerer

3.2.4 Miyazaki (94)

4 Analysis