Assignment 4 – Cost Estimation

Sukhpal Singh
Reg. No. 201788436

Arff files

Attribute-Relation File Format (ARFF) is a file format that holds ASCII text and is used to describe a list of instances with a set of attributes [1]. The Kemerer [2] and Albrecht [3] files were chosen for developing the model. They were chosen because no errors occurred when using Scipy's arff reader. The arff reader formatted the data without any extra modifications needed. One of the errors that arose when reading other files was the @Data attribute was not in the right format for the arff reader, as well as other attribute reading errors.

China and Maxwell files could have been used as well, but there were many more attributes to consider. Whilst the Kemerer and Albrecht data sets only had one effort column, China and Maxwell had several. Therefore, to keep the model simple, Albrecht and Kemerer were chosen.

The Kemerer and Albrecht data files both have the same number of columns, therefore they can be used interchangeably without reprogramming the code. The Kemerer data set has fewer number of entries than the Albrecht. All entries were used either for training or testing. Also, all attributes were used since there was no genuine reason to exclude any of them.

The data file was then passed into a pandas dataframe that could then be used to separate the effort column, and the remaining attributes. The dataframe was split into two separate dataframes, one that held 75% of the dataframe for training, and the other holding 25% of the population for training.

DEAP – Genetic Programming framework

The model was developed using the example found in [4]. It was then modified to fit the assignment.

Firstly, generating trees required a main primitive set to be created using DEAP's creator, and was used to hold user defined primitives. Primitives were then added, consisting of math functions. Maximum, addition, subtraction, multiplication, minimum, cosine and sine were all added to the primitive set. Log, exponential and division were not added because they needed arguments to be predefined, and the aim of the model was to be as general as possible.

Secondly, the fitness had to be created using creator. The fitness was chosen to be minimised, due to the user defined fitness function described later during individual evaluation. An individual tree (IT) then had to be created. This provided information about the characteristics of the IT that had to be created. It was defined to be a primitive tree, with a minimised fitness, and to use the main primitive set for developing functions when making each individual.

After creating the primitive set and the fitness, functions had to be registered to indicate what should be done to the individuals. Components that were registered were added to a toolbox, that could then change the individuals depending on what tools are being requested. An expr function was registered, so that when requested, creates a full tree with a minimum and maximum number of end points. The minimum and maximum values were chosen to be the same, which was the number of columns in the data file, minus one to account for the column holding the effort column. Next, the individuals were made under the title 'individual'. The information passed to these individuals, were the method to make the trees (genFull), the IT creator function and the expr function. Afterwards, the population function was registered. This required to be repeated (initRepeat), to be a list and to use the registered 'individual' creation method. Lastly, to evaluate each tree, required to be compiled. This was a registered function, where compile was called from the genetic programming tools from DEAP.

The next set of tools to be registered into the toolbox were the genetic programming (gp) tools. The gp tools were evaluate, select, mate, exp_mut and mutate. These all were names required for the evolution and could not be changed to user defined names.

The evolution of the trees was then enacted by telling the toolbox to create an n sized population. The method used to evolve the population was eaSimple [5]. The model evolves the population based solely on cross over and mutation [6]. The returned population is independent of the input population [6]. Evaluation was the user defined sub-class, which evaluated the fitness of each individual tree.

Creating the fitness required the compiled tree to replace its' endpoint arguments with values from the training data. The tree was applied to each row of the dataset, replacing the endpoints of the tree with row values, and was summed over to give an overall effort value.

The actual effort of the training data was also summed over. The difference between the actual effort value and the resulting effort, was subtracted and squared, then square rooted to give the overall fitness of each tree. The need to keep the difference positive was related to the minimisation of the fitness, stated when the fitness was created. The smaller the fitness, the smaller the deviation from the actual effort value, therefore more accurate trees were developed. Selection was achieved using tournament selection [7]. Mating/crossover was achieved using cxOnePoint [8]. Mutation was achieved using mutUniform [9]. The mutation method randomly selects a point in the individual tree and swaps it with one generated by the expr function, that creates a genFull tree.

The final population was placed into a pandas dataframe, and complied again to extract the fitness value of each tree, as it was lost when using eaSimple. The returned fitness was then passed to a dataframe. The fitness dataframe was concatenated to the final population dataframe. The returned population usually converged to produce the same individual if there were many generations. The dataframe was sorted in ascending order of fitness, and the first

tree in the dataframe was taken as the final tree/function. Numpy's mean was then applied to return the performance of the given tree/function.

Linear Regression Model (LR)

Scikit learns linear regression method was used for predicting cost efforts [10]. Using the data from the arff file when passed into a pandas dataframe, it was then passed to the cross-validation function. Cross-validation splits the data into two sets, one set for training the data and another for testing the data. The default setting for cross-validation is 75% train and 25% test. This was the same train and test population used for GP modelling.

When passing the dataframes to cross validation/linear regression modelling, one dataframe contained all the parameters except the effort values, whereas the other only contained the effort values. Then the linear regression model was made to fit the dataframes. Once the LR model was developed, it was used to make predictions. The predictions were the cost efforts.

Results

User definable parameters were set to:

```
NumberOfGenerations=100          #ideally choose higher than 50
PopulationSize=60               #ideally choose higher than 50
tournamentsize=3
usercxpb=0.3            #probability of mating two individuals
usermutpb=0.2           #probability of mutating an individual
```

Correction to graphs: The right-hand side is correlated with the red data whereas the left-hand data is correlated with the blue data. The x-axis is always the prediction number, i.e. prediction 0 is the first prediction made using the models.

Albrecht

Trial 1

Figure 1: GP, trial 1, albrecht

Figure 2: LR, trial 1, albrecht

Trial 2

Figure 3: GP, trial 2, albrecht

Figure 3: LR, trial 2, albrecht

GP section: trial 1, albrecht, figure 1

tested
predicted effort mean: 9.79913856241 , actual mean effort: 20.2166666667
Mean difference -10.4175281043
Variance: 0.273327708703

training
predicted train effort mean: 22.4277131925 , actual train efforts: 22.4277777778
Mean difference: -6.45852748704e-05
Variance: 0.482475131615

Linear Regression: trial 1, albrecht, figure 2

Coeffs [-14.72631253 -18.43644126 -14.03007573 -36.92148444 -117.35561135
3.51715031 0.20497262]
Mean squared error 198.437456905
Variance 0.571240891485

GP section: trail 2, albrecht, figure 3

tested
predicted effort mean: -10.5149133551 , actual mean effort: 20.2166666667
Mean difference -30.7315800218
Variance: -5.47134666219

training
predicted train effort mean: 22.4278079924 , actual train efforts: 22.4277777778
Mean difference: 3.02146087812e-05
Variance: -0.469720365779

Linear regression: trial 2, albrecht, figure 4

Coeffs [-30.533041 -38.145581 -29.87356 -76.44438926 -106.98020874
7.519612 0.15357946]
Mean squared error 153.374431352
Variance -0.914176198697

Kemerer

Trial 1

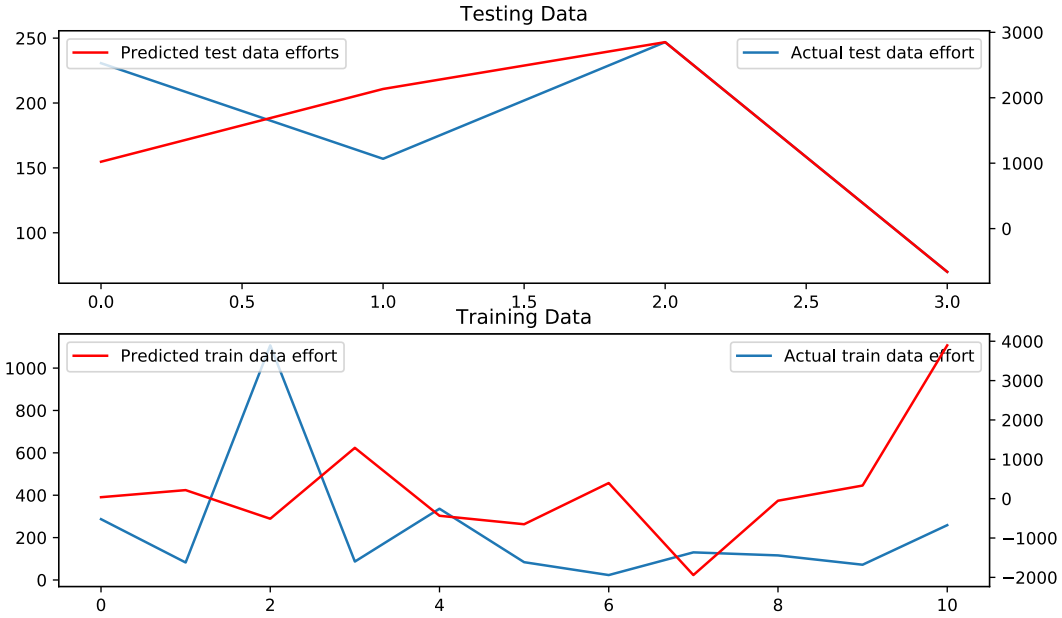

Figure 5: GP, trial 1, kemerer

Figure 6: LR, trial 1, kemerer

Trial 2

Figure 7: GP, trial 2, kemerer

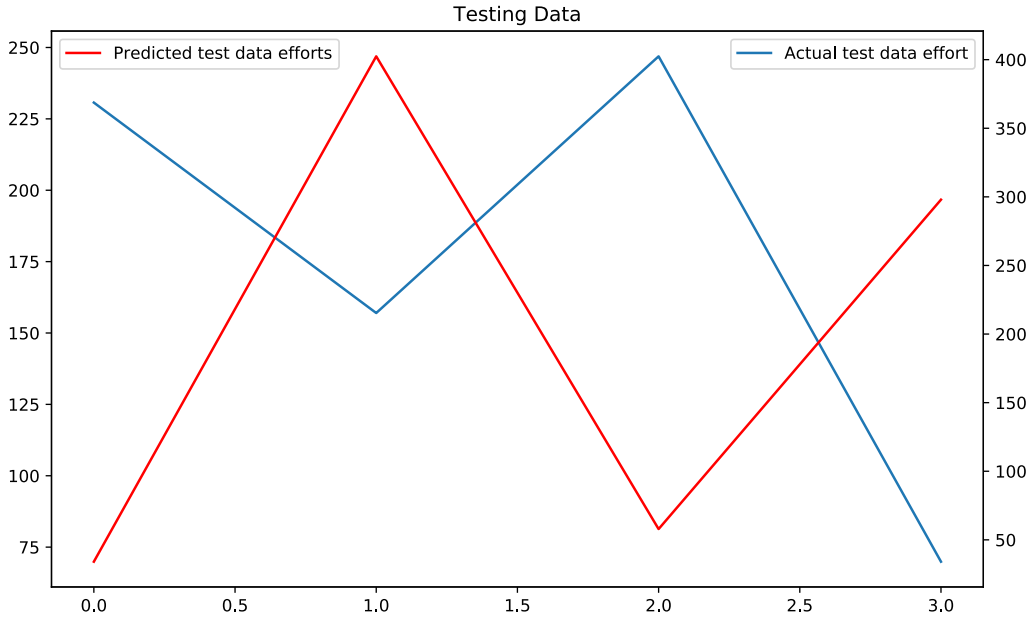

Figure 8: LR, trial 2, kemerer

GP – trail 1, kemerer, figure 5

tested
predicted effort mean: 1336.36684793 , actual mean effort: 176.125
Mean difference 1160.24184793
Variance: -602.259024203

training
predicted train effort mean: 234.727943843 , actual train efforts: 234.928181818
Mean difference: -0.200237974734
Variance: -0.0867737384832

Linear Regression: trial 1, kemerer, figure 6

Coeffs [-4.80818852e+01 1.05658785e-12 1.02880881e+02 1.07462802e+01
-1.63640151e+00 -1.45431039e-01 9.73720195e-01]
Mean squared error 131613.01539
Variance -243.685307463

GP – trial 2, kemerer, figure 7

tested
predicted effort mean: 0.824410175433 , actual mean effort: 176.125
Mean difference -175.300589825
Variance: -6.25886087975

training
predicted train effort mean: 234.929007023 , actual train efforts: 234.928181818
Mean difference: 0.000825204553166
Variance: -1.28732695567

Linear Regression: trial 2, kemerer, figure 8

Coeffs [-20.88273962 -35.75124055 96.90721819 -7.9412839 -1.8139539
-0.41455916 1.19694544]
Mean squared error 32809.4219377
Variance -2.69440005199

The intention of plotting the model onto the training data, as well as the testing data, was to determine whether the model reproduced the effort values that were used to help develop the model. Visible in figures 1 and 3 show that the better the model is at replicating the training efforts, the better at predicting future test efforts. This is reflected in the mean difference. The mean difference gives the resultant between the predicted effort and the actual effort. The polarity of the difference indicates the direction the predicted mean is located in reference to the actual mean, i.e. if the predicted mean was larger (positive) or smaller (negative) than the actual mean. However, irrespective of the polarity, the larger the difference the less accurate the model. Hence, if the model cannot reproduce the training data well, then it is less likely to predict future effort values accurately. Therefore, a smaller mean difference, the more biased towards the training data, the more likely to create accurate predictions.

Another distinguishable trait between figures 1 and 3 is that the evolved model is not consistent. Whilst one model may produce a function that reproduces the shape of the actual effort values, the values themselves may not be accurate (y-axis values/effort values). This is consistent with the implication of what the mean indicates about the quality of the model.

Surprisingly, the LR model shown in figures 2 and 4, predict responses that are opposite in amplitude, i.e. instead of a trough a peak was predicted etc. This could be the result of there being a limited number of entires used for testing. The only case that shows the LR model producing any sort of similarity in the shape of the effort curve is in figure 6.

Conclusion

Linear regression was used for comparison against genetic programming for effort estimation. Despite the problem seeming to be more linear problem at first glance, the LR results indicate that using LR is in fact the less accurate choice. Whereas, the GP model showed it was more reliable at predicting effort values. Applying the developed function onto the training data was useful to establish an estimation of accuracy for the created model. The accuracy could also be found using the mean difference between predicted efforts and actual efforts.

References

[1] Attribute-Relation File Format, [last viewed: 27/11/17], available at: https://www.cs.waikato.ac.nz/ml/weka/arff.html
[2] Kemerer, Jacky W. Keung, [last viewed, 27/11/17], available at: https://zenodo.org/record/268464#.WggyXbCFi_A
[3] Effort Estimation, Yanfy Li, Jacky W. Keung, [last viewed, 27/11/17], available at: https://zenodo.org/record/268467#.WggyiLCFi_A
[4] Symbolic Regression Problem: Inftroduction to GP, [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/examples/gp_symbreg.html
[5] Algorithms, [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/api/algo.html#module-deap.algorithms
[6] Algorithms, varAnd, [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/api/algo.html#deap.algorithms.varAnd
[7] Evolutionary Tools, selTournament , [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/api/tools.html#deap.tools.selTournament
[8] Evolutionary Tools, cxOnePoint, [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/api/tools.html#deap.gp.cxOnePoint
[9] Evolutionary Tools, mutUniform, [last viewed, 27/11/17], available at: https://deap.readthedocs.io/en/master/api/tools.html#deap.gp.mutUniform
[10] sklearn.linear_model.LinearRegression, [last viewed, 27/11/17], available at: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html