Genetic programming-based construction of features

Dino Drokan Piero Battelli Ivan Buterin

Abstract— **The main goal of this project was to improve the prediction accuracy of the machine learning model, using genetic programming, by building new sets of features out of already existing features in a given dataset. First it was necessary to implement the standard method and compare the results to the initial prediction given by the machine learning algorithm, in this case the random forest model. Afterwards, the extended method is implemented by splitting up the original features into evolving features and hidden ones. The evolutionary algorithm generated better features which slightly improved prediction quality. At the end of the paper the results are being discussed and elaborated upon.**

I. INTRODUCTION

The basic premise of this project was constructing features for machine learning (ML) inducers using genetic programming in order to examine the possibility of generating better ML models. Newly constructed features, rather than old ones, are being sent to the ML model inducer. Constructed features are actually generated using the original (initial) ones where each new feature is computed by a mathematical expression involving values of original features. Therefore the value of the new features is computed using values of multiple original features.

In the following chapters, there will be given a detailed explanation of the standard method and the extended method which splits the original dataset in two parts. The reasoning behind this was to compare the results and efficiencies of both methods. During the making of this project, the scientific paper named "Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks" by Krzysztof Krawiec has been used as reference and guidance. Through trial and error, the random forest model has been selected as the machine learning classifier implemented in this project. The original datasets will be expanded upon in the coming chapters, but they are based around the features of peptides.

II. METHODOLOGY

As the project is using a standard genetic programming template with slight modifications, the majority of attention was dedicated on developing the machine learning code first. The machine learning part is used to calculate the fitness scores of each individual which make up a population. The random forest model is being used because it can handle large datasets with ease and it provides a larger accuracy in predicting outcomes regarding the precision tree algorithms. The Matthew's correlation coefficient (MCC) is being used as the quality measure of the machine learning predictions. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. In statistics it is also known as the phi coefficient.

Before the evolutionary run, the original data set is split into the training dataset and the test dataset. The training dataset will be used as part of the evolutionary run and the test dataset will be used outside of the algorithm. The population is made out of 500 individuals, each individual being represented by 10 different features. Every single feature is represented by a tree data structure, where the value of each node is either a mathematical function or column name (terminal) of the original dataset.

Column names are only used as the values of the terminal nodes. The set of mathematical functions consists of: addition, difference, multiplication, division, less than (returns smaller value), greater than (returns greater value), equal (if two input variables are equal, returns 1, if not returns 0). These functions have two values as input and produce one value as output.

The binary tree is a data structure used to represent one feature. Therefore, the value of a new feature, for a specific row, is calculated by recursive calculation of the binary tree. Figure 1 shows the definition of one feature. It uses 2 mathematical operations: addition and multiplication. The remaining nodes are terminal nodes with column names as values.

When a value of a feature for a certain row (row inside of the original dataset) is being calculated, column names are changed with exact values from the original dataset. The value is then calculated recursively.

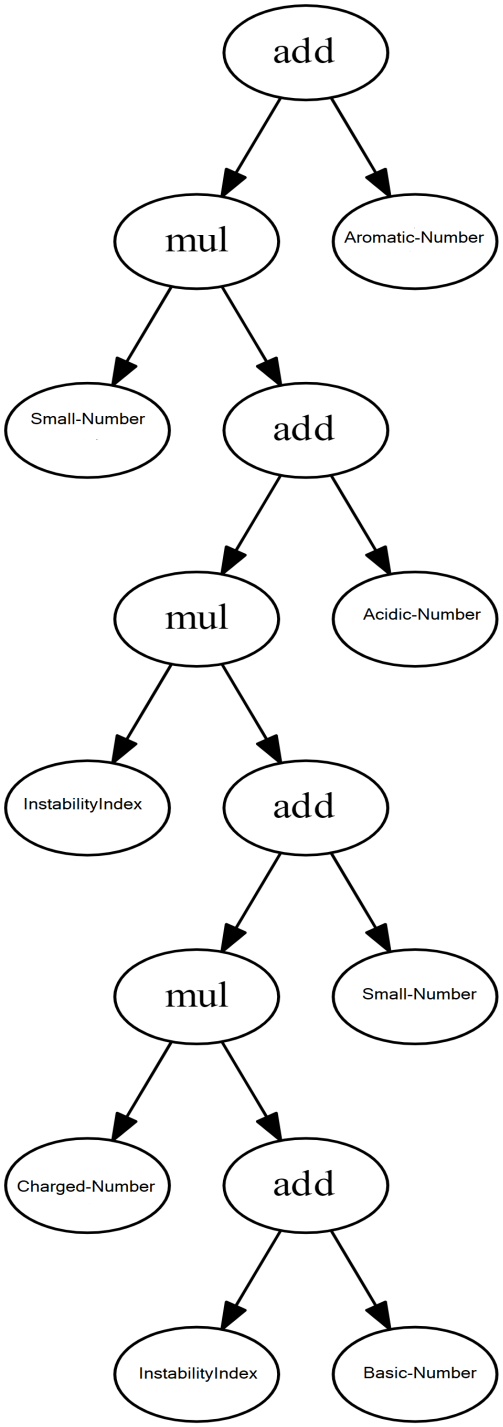

Fig. 1. Binary tree representing one feature

In the end every single individual has its own specific dataset with its own specific set of features. The dataset of the individual is generated from the original dataset. Therefore each individal is defined by: a list of 10 features (trees), a dataset (constructred by features), the fitness value (MCC value calculated by fitness function), a test dataset (constructed from test data), the fitness score obtained from test data (MCC value for test data), ML classifier, indices of evolving features and indices of hidden features.

A random population is generated and goes through an evolutionary run to create offsprings and hopefully improve the fitness (MCC) of the individuals inside the population. As with any genetic program, it is important to first do the crossover followed by mutation. The individuals that will take part in the crossover are being selected by the tournament selection algorithm, meaning that the best individuals will participate in the crossover. The tournament selection algorithm selects, at random, a defined number of individuals. Then it returns the individual with the best fitness score.

The crossover function simultaneously iterates through features (trees) of both parents. Then for each pair of features, a randomly selected subtree belonging to the second parent is added to a node of the first parent. A subtree from the first parent replaces a subtree in the second parent. Therefore, a new individual is constructed.

The new individual then goes through the mutation process in which the code checks every single node of a given tree and if the randomly generated number is less than the defined mutation probability (set to 15 percent in this project) a new, randomly generated subtree is added to that node.

The crossover function, with mutation, results in a new set of individuals (offsprings) which are then added to the population. The population is now a mixture of the old population and newly generated offsprings as a result of the crossover and mutation functions.

Afterwards, the new population is sorted by fitness scores in a descending order and the first 500 (population size) individuals are selected as the new generation of the population.

III. CASE STUDY

There were two main datasets used as part of the development and testing of the implemented code. The former being a smaller scale dataset (1024 rows) while the latter was of much greater size (9409 rows) and used only once the project was already in a working state. Both datasets contain data surrounding peptides, the former those that have antiviral features while the latter is focused on antimicrobe peptides. Every single peptide is made out of aminoacids and their sequencing is shown in the column called sequence. The peptides with a label of the value 1 have the antiviral (or antimicrobe) feature while those with the value 0 do not. The rest of the features represent the physiochemical features of the peptides. Using these two original datasets the general idea was to make a new dataset that would improve the prediction accuracy of the random forest model. The dataset was split into three smaller datasets using the built in functions of the SKLearn library. The three new datasets were the training (80 % of data), validation (10 % of data) and testing (10 % of data) datasets. The training data was used in the training of the machine learning algorithm

while the validation data was used during the development process as a means of testing the current ML models. The test data was used as a means of calculating another MCC score once a new generation of offsprings has already been generated. Initially, accuracy was used as the main quality indicator but given that it has some issues when the datasets are unbalanced, meaning that there's way more data of one type compared to the rest, the indicator being used by the end of the project was the Matthew's Correlation Coefficient also known as MCC. Before going through an evolutionary run the initial MCC value of the smaller dataset used as part of this project was between 0.5 and 0.6. Concerning the two different methods used it is important to note that both the standard and extended approach use a population made out of 500 individual and the number of generations is set to 50. The standard method is using all of the said features while the extended method splits the original features into two sets, the evolving and the hidden feature set. The evolving set consists of 7 separate features that always take part in the evolutionary run while the hidden features consist of 3 features of the highest fitness scores that will be kept safe as long as the evolutionary run is happening and no feature of a great fitness score has been constructed. If a newly generated feature has a greater fitness score than one of the hidden features, they swap places.

IV. RESULTS

Figure 2 shows fitness scores of the 12th generation when code was run with smaller dataset. At the 12th generation program stopped executing because value of certain feature was too big to fit inside float datatype. The initial MCC value for the original dataset is calculated before evolutionary run.

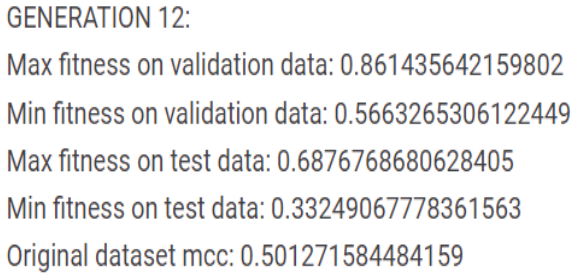

Fig. 2. Fitness scores for the 12th generation (small dataset)

The results show significant improvement in the quality of the prediction on the smaller scale dataset which is influenced by newly generated features. The original MCC score of approximately 0.5 increased to 0.86 using the validation dataset while the test dataset resulted in an MCC score of 0.76. Due to time restriction it wasn't possible to optimise the algorithm on the bigger dataset which resulted in less of a difference in the MCC score compared to the original. Code was stopped at thirty-third generation due to variable overflow of certain feature. Evolutionary parameters were modified for bigger dataset because execution was too slow. Therefore code ran with population size of 100 and offspring size of 20.

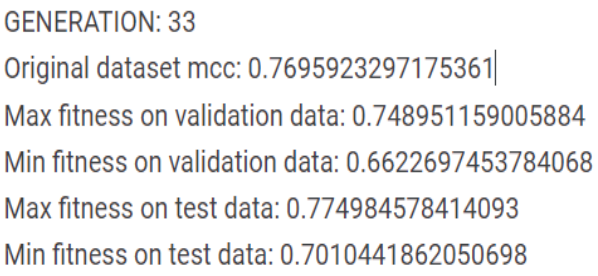

Fig. 3. Fitness scores for the thirty-third generation (big dataset)

The result of the program, for a certain dataset as input, is a new set of features with ML classifier which then can be used to predict values for new data.

V. DISCUSSION

As expected, the return value, being the MCC, was greater when using the extended approach compared to the standard method. The standard method allows all features to take part in the evolutionary run and by doing so there's a chance that some features that are already good might become worse after further modifications. On the other hand, the extended approach hides the best features generated thus far and protects them from further modifications unless there's a new feature that's better than a hidden one which is when they'd lose their place and return in the evolving dataset that takes part in the crossover and mutation. The standard method usually resulted in an MCC value between 0.6 and 0.7 which is better than the MCC value we've had before going through the evolutionary run and it is trending towards +1 which would be a perfect prediction but it is still a worse value than the one generated by the extended approach which resulted in an MCC value between 0.8 and 0.9. It is also important to note that it has been noticed that when using the standard approach sometimes the fitness score would decrease from one generation to another because of the fact that a good feature would get modified and hence "lost". That's not the case with the extended method because the good features are protected and every new generation has a fitness score equal to or greater than the generation which it comes after.

VI. CONCLUSION

The main goal of this project was to improve the results of the machine learning model predictions. The MCC value increased from a range between 0.5 and 0.6 to a value of 0.8 to 0.9 using the extended approach on the smaller dataset. The bigger dataset could still use some optimisation and testing to reach its full potential and increase the MCC score of the new dataset even further. It might be possible that an even better MCC value could be achieved by experimenting with genetic program parameters such as: number of generations, population size, mutation probability, offspring size, number of competitors in the tournament selection. Additionally,

using different machine learning algorithms and comparing them to each other is also a viable option for further testing. The code also runs fairly slow when the larger dataset is used so finding ways to optimise the running time is something to keep in mind as well if someone were to further develop the project.