Rice Classification (COSC 4P82)

Kelvin Odinamadu*, Gideon Oludeyi†
Computer Science
Brock University
St. Catharines ON, Canada
*ko20so@brocku.ca †go21zq@brocku.ca

***Abstract*—This report explores the application of genetic programming (GP) for the classification of rice varieties [1], leveraging a dataset to train and evaluate the fitness of our solutions. By employing genetic programming, we aim to improve classification accuracy. This report not only enhances our understanding of GP's application on classification tasks, but also sets a precedent for future studies aiming to optimize crop classification and management practices.**

***Index Terms*—Genetic Programming, Rice Classification, Agricultural Technology, Machine Learning, Data Analysis**

I. Introduction

In agricultural advancements, the classification of rice varieties is a pivotal challenge, significantly impacting yield and quality assessments. This study harnesses the power of genetic programming (GP), to develop a sophisticated model capable of classifying rice varieties with high precision. Leveraging the DEAP Python library [2], our experiment aims to explore the efficacy of GP in navigating the complex feature space of rice data to predict a variety of classifications accurately.

II. Background

Genetic Programming consists of several sub-algorithms that are used to evolve a computer program to solve a particular problems. The choice of algorithms impacts the effectiveness of finding an approximately optimal solution. The following types of algorithms together make up the main portion of Genetic Programming: selection algorithms, crossover operators, and mutation operators.

A. Selection Algorithm

Selection algorithms are responsible for choosing individuals whose characteristics will be propagated to future generations. The mainstream selection algorithms that are usually used in GP are Roulette Wheel selection and Tournament selection. We decided to use Tournament selection for this classification task as it is easy to implement. The process of tournament selection works by randomly choosing k individuals from the current population and selecting the individual with the best fitness of those k individuals to proceed to the mating pool for reproduction. This process is repeated until enough individuals have been selected for reproduction (ie. population size is reached).

B. Crossover Operator

The role of crossover operators is to simulate the process of inheritance found in nature, whereby new offsprings are produced and exhibit similar characteristics to their parents.

We used the One-Point crossover operator which takes a random sub-tree each from two parent individuals, swapping them to produce two new offspring individuals for the next generation. This crossover operator, implemented in DEAP [2], is referred to as `cxOnePoint`.

C. Mutation Operator

As found in nature, mutations in GP simulate random modifications of certain characteristics of an individual and are not inherited from parents. Mutations encourage a GP algorithm to explore more of the search space to uncover potential areas of improvements.

We used the Uniform mutation operator which works by replacing a random sub-tree of the individual with a generated tree. This mutation operator is implemented in DEAP [2] as `mutUniform`.

III. Experimental Setup

The GP system of the DEAP [2] Python package was used for the experiment. The experiment consisted of 10 runs over the Rice (Cammeo and Osmancik) dataset [1] with each run initialized with a different seed for the random number generator, and the results were averaged up.

A. Dataset

The Rice (Cammeo and Osmancik) dataset [1] consists of 7 features obtained from 3810 images of rice, and binary labels indicating the type of rice associated with the specified features.

TABLE I
Variable Definition

Variable	Role	Type
Area	Feature	`int`
Perimeter	Feature	`float`
Major_Axis_Length	Feature	`float`
Minor_Axis_Length	Feature	`float`
Eccentricity	Feature	`float`
Convex_Area	Feature	`int`
Extent	Feature	`float`
Class	Label	`"Cammeo" or "Osmancik"`

Prior to training, the dataset was randomly shuffled and 80% were placed into the training set and the remaining 20% into the testing set. The training set was used to evaluate the fitness of the function and drive the generations to better solutions. The testing set was used to evaluate the solutions on unseen data to determine how well it generalizes to classifying new rice instances it has never seen.

B. Language

A GP program needs to define a language for the set of functions it can utilize and manipulate to solve a given problem. In the case of classifying a list of numerical features as either ”Cammeo” or ”Osmancik” rice, we include several arithmetic operators in our language as seen in Table II.

The output of a solution yields a single floating-point number y, which we interpret as the label ”Cammeo” when $y < 0.0$, and the label ”Osmancik” when $y \geq 0.0$.

TABLE II
GP LANGUAGE

Parameter	Definition	Arity
add	x + y	2
subtract	x - y	2
multiple	x * y	2
negative	-x	1
Ephemeral Constants	$x \in (-1.0, 1.0)$	0

C. Fitness Function

The fitness function for this classification task executes a GP solution function on each example in the training set and reports the fitness as the number of hits over the total number of training examples. In other words, for the training set T, the fitness of individual i is defined as $f_T(i) = \frac{hits_i(T)}{|T|}$ where $hits_i(T)$ is the number of hits on the training set for individual i. This fitness value is the accuracy of the individual on the training set.

However, for individuals with tree depth of 15 and higher, we report a fitness value of 0.0 to mitigate bloat in the solutions and encourage a search for smaller yet accurate solutions. The pseudo-code in Algorithm III-C describes the implemented fitness evaluation function from a high-level. A few optimizations were made, however, to the fitness evaluation of an individual but the resulting fitness value remains the same.

TABLE III
GP PARAMETERS

Parameter	Value
Population Size	1000
Max. Tree Size	15
Tournament Size (k)	3
Max. Generations	100
Crossover Rate	0.8
Mutation Rate	0.2
Elitism	2
Max. runs per experiment	10

Algorithm 1 Fitness evaluation algorithm

```
fitness ← 0.0
if individual ≥ 15 then
  fitness ← 0.0
else
  hits ← 0
  for all example in training set do
    prediction ← individual's prediction of example
    if prediction ≥ 0.0 AND label = 1.0 then
      hits ← hits + 1.0
    else if prediction < 0.0 AND label = 0.0 then
      hits ← hits + 1.0
    end if
  end for
  fitness ← hits / training set size
end if
return fitness
```

IV. EXPERIMENT RESULTS

A. Performance of Runs

Figure 1 shows the trend of the average and best population fitnesses on both the training and testing set. Although the GP training process does not interact with the testing data, we see that the accuracy of the solutions on the testing set is almost the same as with the training set. This shows that the solution trees provide a good approximation for classifying unseen instances of rice data.

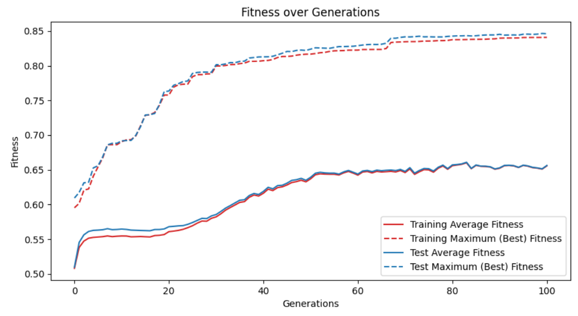

Fig. 1. Average convergence of the average and best fitness over time

Table IV offers another view on the performance of the solution trees. On both the training and testing data, we see that the average and best fitness values of final solutions across runs are essentially identical. This further supports the fact that the solutions are learning the representation of ”Cammeo” rice from ”Osmancik” rice, and are not simply overfitting to the training data. It is worth noting again that the solutions are not trained on testing data throughout the training process.

Table V shows the result of the best solution tree across all runs. We see that the solution is able to accurately predict ”Osmancik” rice than ”Cammeo” rice. This is due to the fact that there are more rice instances labeled ”Osmancik” than instances labeled ”Cammeo”. However, we see that both the

TABLE IV
SUMMARY OF RUNS

	Training	Testing
Average	84.08%	83.82%
Best	93.08%	93.70%

false negative and false positive values are quite low, summing up to a 6.30% error.

TABLE V
CONFUSION MATRIX (TEST SET)

		Actual	
		Cammeo	Osmancik
Predicted	Cammeo	35.17%	3.28%
	Osmancik	3.02%	58.53%

V. CONCLUSION

Genetic Programming (GP) has proven to be an effective tool in classification tasks such as the rice classification problem. Leading off of the Rice (Cammeo and Osmancik) dataset [1], we defined a language for the GP problem and evolved solutions across generations to approximate a good classification model for the classification task.

In the future, we hope to expand on the experiments performed here and incorporate a form of feature engineering to reduce the dimensionality of the problem. One approach we foresee is running GP system on all combinations of independent variables to determine which variables correlate strongly with target label. By identifying the strongly correlated variables, we can discard the other independent variables and potentially reach a better solution.

VI. APPENDIX

Best solution across runs:

```
  add(mul(add(Extent,
mul(mul(Eccentricity, mul(Eccentricity,
-0.006921542832986072)),
add(Minor_Axis_Length,
sub(Minor_Axis_Length,
Convex_Area)))), add(Major_Axis_Length,
Major_Axis_Length)), mul(Perimeter,
neg(neg(add(mul(sub(Major_Axis_Length,
Minor_Axis_Length), add(Minor_Axis_Length,
mul(Convex_Area, -0.006921542832986072))),
add(Perimeter, 0.8851279664759963))))))
```

REFERENCES

[1] Rice (Cammeo and Osmancik). (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5MW4Z.

[2] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary algorithms made easy,” Journal of Machine Learning Research, vol. 13, pp. 2171–2175, jul 2012.