“Genetic Programming for Automatically Evolving Multiple Features to Classification” *Online Supplementary Materials*

Peng Wang[a,b], Bing Xue[b], Jing Liang[c,a], Mengjie Zhang[b]

[a]*School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[b]*Center for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand*
[c]*School of Electrical Engineering and Automation, Henan Institute of Technology, Xinxiang 453000, China*

1. Introduction

This is the Online Supplementary Materials of “Genetic Programming for Automatically Evolving Multiple Features to Classification”.

1.1. Explanation on the Representation

In the manuscript, we use the two data types, i.e., *list* and *array*, as examples to show different types of *concat* functions. As shown in Fig. 1, three types of *concat* functions will be used according to different types of inputs. Specifically, the four functions $+, -, \times$, and the protected $\div$ are used to construct new high-level features. The data types of the input and output of the four functions are all set as *list*. In addition, the data types of all features in a dataset are also set as *list*. Only the data type of output of one *concat* function is an *array*. This is to ensure that the outputs of *concat* function are not used for the $+, -, \times$, and the protected $\div$ functions. The reasons are shown below, which are also the main motivations of this manuscript using the strongly-typed GP.

For simplicity, a simple case with $D = 10$, which represents 10 features, $f_1, f_2, \ldots, f_{10}$, in a dataset, is considered here. As shown in Fig. 2, for the function $\div$, its left sub-tree has three outputs $\{f_2, f_9, f_3\}$ while its right sub-tree has two outputs $\{f_5, f_8\}$. If the length of the output vector of each function is limited to the lower length between two input vectors. The constructed features from Fig. 2 can be $\{f_2/f_5, f_9/f_8\}$. Although

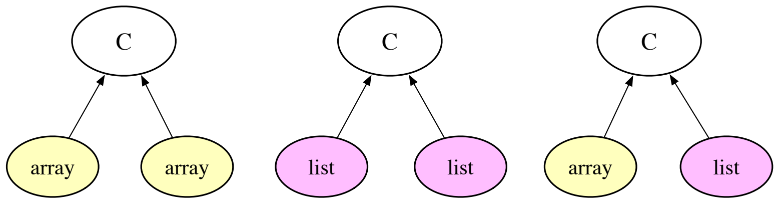

Figure 1: Three types of *concat* functions.

a GP method with this vector representation can have multiple outputs, redundancy across the tree may affect the efficiency. That is because some features represented by circle with light orange color such as f_3 in Fig. 2 are in the tree, but they are not involved in the final output.

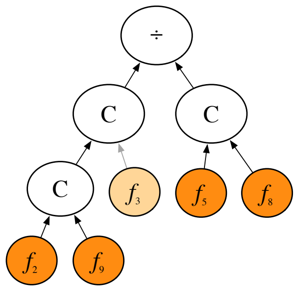

Figure 2: Example of the GP representation.

1.2. Analysis on the Bottom-Up Pruning Technique

Fig. 3 is used to clarify the pruning process. On the left of Fig. 3, nodes with gray color mean that the resulting features are duplicated ones, i.e., the same features are selected and/or constructed in the tree. According to the bottom-up order, C_3, C_2, C_4, C_1 will be checked one by one. The function C_2 will be replaced by function C_3 since f_3 is already in U_f after checking C_3. At this stage, $U_f = \{f_1, f_3\}$. When checking C_4, two same features are created, i.e., $f_2 \times f_5$ and $f_5 \times f_2$. Therefore, only one feature from the two will be kept, and $f_2 \times f_5$ will replace the function C_4. Finally, the proposed BUP technique can obtain a reduced tree as shown on the right of Fig. 3. Although the process involves structural changes, these modifications will not alter the semantics. Therefore, the proposed BUP operator can preserve the semantics of

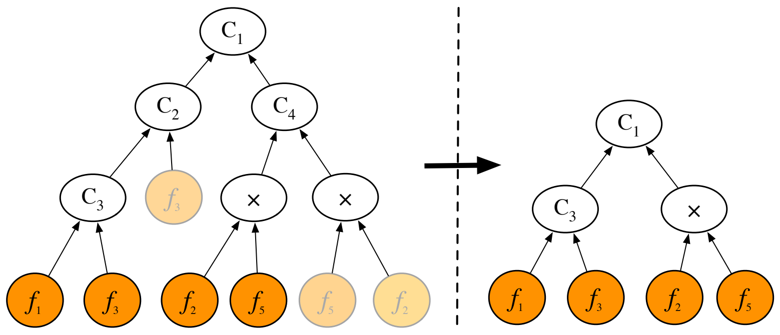

Figure 3: GP representations and performing the bottom-up pruning technique.

individuals while removing redundant parts.

The computational cost of the bottom-up pruning mechanism is typically associated with evaluating and deciding which nodes or subtrees to prune (i.e., tree traversal), which is mainly influenced by the number of $contat$ functions included in a tree. Let n_c means the number of $contat$ functions that are candidates for pruning. The total computational complexity of the bottom-up pruning mechanism is between $O(n_c)$ and $O(2n_c^2 - n_c)$. The reasons are om the following:

The outputs of each $contat$ function include two features. If they are duplicated, only one feature will be kept. In other words, only one sub-tree will be retained, and the other one will be deleted from the tree. Considering the following two cases. Suppose that all features output from the n_c $contat$ functions are the same. Then, there will consume n_c comparisons. If all features output from the n_c $contat$ functions are different, there will be consume $n_c * (2n_c - 1)$ comparisons. The reasons are as follows. The first $contat$ function consumes 1 comparison, the second one consumes 5 comparisons (the left/right output is compared twice with the two outputs from the first $contat$ function, and plus one comparison between the two outputs from the current $contat$ function), the last one consumes $4 * n_c - 3$ comparisons, thus the overall is $1 + 5 + \cdots + (4 * n_c - 3) = n_c * (2n_c - 1)$. Therefore, the overall computational complexity of the bottom-up pruning mechanism is between $O(n_c)$ and $O(2n_c^2 - n_c)$.

1.3. Analysis on the Proposed SCGP Method with SVMs

Further experiments using SVM with radial basis kernel function as the leaning algorithm have been added. The average test classification accuracy and the average dimensionality obtained by the methods are shown in Table 1 and Table 2, respectively. In addition, the Wilcoxon test with a significance level of 0.05 is used to test whether there is a statistically significant difference between the proposed SCGP method and other algorithms. The signs ‘↑’, ‘↓’, and ‘≈’ indicate that the corresponding benchmark algorithm is significantly better than, worse than, and has no significant difference from SCGP, respectively.

Table 1: Average classification accuracy (%) of different methods on the test sets. The highest test accuracy obtained on each dataset is in bold.

Dataset	FULL	PCA	GRP	SRP	FCBF	SBMLR	VGP	MGP	SCGP (ours)
WBCD	91.23↓	90.64↓	87.37±5.31↓	86.63±6.44↓	94.15↓	94.15↓	96.10±1.04≈	96.00±1.17≈	**96.57**±0.69
Ionosphere	92.45↓	92.45↓	91.10±1.78↓	90.28±1.60↓	92.45↓	87.74↓	89.34±2.59↓	86.42±3.12↓	**93.11**±1.01
Movement	76.85↓	74.07↓	60.43±10.47↓	62.72±9.43↓	73.15↓	**78.70**≈	68.55±4.81↓	65.46±2.72↓	78.64±1.76
Hillvally	48.35↓	47.80↓	47.63±1.15↓	47.53±1.38↓	47.25↓	48.63↓	97.27±12.66↓	99.45±0.38≈	**100**±0.0
Musk1	97.38≈	97.21≈	92.06±1.57↓	92.14±1.58↓	74.75↓	87.38↓	92.83±1.86↓	93.43±1.23↓	**97.75**±0.74
Arrhythmia	57.35↓	59.56↓	54.24±1.18↓	54.90±1.53↓	63.24≈	58.09↓	62.08±2.44≈	62.23±3.49≈	**63.33**±2.72
Darwin	58.49↓	58.49↓	57.67±6.51↓	53.14±9.51↓	77.36↓	77.36↓	78.62±4.76↓	78.74±3.61↓	**82.64**±3.39
Madelon	67.05↓	69.10↓	48.73±0.51↓	48.77±0.76↓	61.15↓	69.36↓	61.26±1.61↓	60.45±1.22↓	**84.63**±1.01
QSAR_rogen	93.69↓	93.89≈	93.45±0.26↓	93.37±0.28↓	**95.86**↑	93.49↓	93.76±0.88≈	94.44±0.65≈	94.14±0.71
AD	93.60↓	93.39↓	92.95±1.52↓	87.80±0.73↓	94.21↓	92.07↓	96.04±0.55↓	96.34±0.82↓	**97.21**±0.37
Leu1	68.18↓	95.45↑	65.30±9.28↓	64.55±11.16↓	**95.45**↑	72.73↓	85.91±6.78≈	85.30±8.52≈	85.76±8.76
Leu2	54.55↓	81.82↓	54.24±9.46↓	53.18±9.62↓	77.27↓	**90.91**↑	80.15±7.56↓	75.61±11.41↓	84.09±7.76
DrivFace	96.15≈	97.25↑	90.86±0.56↓	90.79±0.37↓	**97.80**↑	**97.80**↑	94.87±1.85↓	93.42±3.47↓	96.56±1.05
ALL	68.18↓	90.91≈	68.33±7.09↓	67.12±6.18↓	81.82↓	**95.45**↑	81.82±11.79↓	72.42±10.23↓	89.70±5.86
CLL	55.88↓	47.06↓	53.24±7.61↓	52.35±7.53↓	64.71↓	55.88↓	**68.41**±7.69≈	63.63±8.84↓	67.16±9.33
TCGA	98.76↓	99.17≈	98.63±0.37↓	98.66±0.30↓	99.59≈	99.17≈	95.52±2.10↓	96.36±1.94↓	**99.88**±0.67
Sum	2≈,14↓	2↑,4≈,10↓	16↓	16↓	3↑,2≈,11↓	3↑,2≈,11↓	5≈,11↓	5≈,11↓	N/A
Rank	5.36	4.29	7.71	8.29	3.89	4.57	4.18	4.79	1.93

As shown in Table 1, the proposed SCGP method achieves significantly better test classification performance than the eight baseline methods. Only on two datasets (Musk1 and DrivFace), SCGP shows similar test accuracy to FULL, but SCGP significantly reduces the dimensionality of the obtained features which can be seen in Table 2. Moreover, the proposed SCGP method obtains significantly lower dimensionality than PCA, GRP, and SRP on six datasets (Ionosphere, Leu1, Leu2, DrivFace, ALL, and

Table 2: Average dimensionality of the obtained features. The smallest size obtained on each dataset is in bold.

Dataset	FULL	PCA/GRP /SRP	FCBF	SBMLR	VGP	MGP	SCGP (ours)
WBCD	30↓	**1**↑	5↑	5↑	4.4±1.7↑	7≈	7.2±1.8
Ionosphere	34↓	17↓	9↓	5↓	**2.1**±1.7↑	7↓	4.8±1.9
Movement	90↓	**7**↑	9↑	66↓	15.7±3.8↑	**7**↑	19.4±4.0
Hillvally	100↓	**1**↑	**1**↑	2≈	2.4±1.3≈	7↓	2.3±1.3
Musk1	166↓	18↑	**3**↑	10↑	11.0±4.6↑	7↑	27.9±3.7
Arrhythmia	279↓	26↓	8↑	**4**↑	11.3±3.8↑	7↑	12.7±6.0
Darwin	450↓	**1**↑	30↓	12↑	5.0±2.4↑	7↑	16.4±4.2
Madelon	500↓	223↓	**2**↑	15↓	3.2±1.6↑	7↑	8.7±3.2
QSAR_rogen	1024↓	304↓	19↓	2↑	**1.8**±1.2↑	7↓	5.0±3.1
AD	1558↓	**2**↑	41↓	4↑	2.7±2.5↑	7↑	19.4±6.3
Leu1	5147↓	26↓	41↓	5↓	**1.3**±0.5↑	7↓	2.7±0.7
Leu2	5327↓	27↓	36↓	14↓	**2.5**±0.7≈	7↓	2.9±1.0
DrivFace	6400↓	45↓	33↓	50↓	8.2±3.4↑	**7**↑	10.3±3.4
ALL	7129↓	27↓	51↓	17↓	**4.5**±1.5↑	7↓	5.3±1.3
CLL	11,340↓	**4**↑	70↓	11↑	4.5±1.9↑	7↑	12.1±3.6
TCGA	20,531↓	237↓	3277↓	29↓	18.5±2.1↓	**7**↑	14.9±3.3
Sum	16↓	7↑,9↓	6↑,10↓	7↑,1≈,8↓	13↑,2≈,1↓	9↑,1≈,6↓	N/A
Rank	7.00	3.89	4.32	3.61	2.07	2.96	4.14

TCGA), while PCA achieves significantly better test classification performance than SCGP only on the Leu1 and DrivFace datasets. Furthermore, SCGP outperforms FCBF and SBMLR on eight datasets with approximately 10% or even more improvements on three datasets (Hillvally, Musk1, and Madelon) in terms of the test classification accuracy. In addition, as shown in Table 1 and Table 2, both VGP and MGP fail to achieve significantly better classification performance than SCGP, although VGP obtains better dimensionality on 15 datasets than SCGP.

Overall, the proposed SCGP method with SVMs can reduce the dimensionality of the used datasets to less than 28 features on average while still maintaining a quite good classification performance. Although different learning algorithms (KNN and SVMs), are used, the proposed SCGP methods can still construct new high-level features to help the learnt model significantly improve the classification performance, respectively.