Late-Acceptance and Step-Counting Hill-Climbing GP for Anomaly Detection

Van Loi Cao
University College Dublin
Dublin, Ireland
loi.cao@ucdconnect.ie

Miguel Nicolau
University College Dublin
Dublin, Ireland
miguel.nicolau@ucd.ie

James McDermott
University College Dublin
Dublin, Ireland
james.mcdermott2@ucd.ie

ABSTRACT

One-Class Classification (OCC) for anomaly detection is a method for anomaly detection that constructs a classifier from only normal examples. Classifier systems such as Kernel Density Estimation (KDE) and Support Vector Machine (SVM) typically do well at this task, but can be slow when classifying new instances. Previous work has used Genetic Programming (GP) to learn the density from KDE, with results often out-performing those of one-class SVM (OCSVM) and KDE based OCC (OCKDE). However, the search is computationally expensive, and it suffers from a need to tune many parameters. In this paper, we will introduce the Late Acceptance Hill-Climbing (LAHC) and Step Counting Hill-Climbing (SCHC) algorithms as GP alternatives. These are simple hill-climbing algorithms, with specific methods to avoid local optima, and far less parameters to tune. The results demonstrate that the proposed models are competitive with standard GP, and often out-perform OCSVM. Their query-time is much less than that of OCSVM, and does not scale with the size of training data.

CCS CONCEPTS

•Security and privacy → Intrusion/anomaly detection and malware mitigation;

ACM Reference format:
Van Loi Cao, Miguel Nicolau, and James McDermott. 2017. Late-Acceptance and Step-Counting Hill-Climbing GP for Anomaly Detection. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3076091

1 INTRODUCTION

One-class classification is a common approach for anomaly detection, especially in the network security domain [1, 5]. Recent approaches have applied GP and artificial neural networks for this task [4–7]. In previous work [5] we employed standard GP to learn the density function produced from KDE. A one-class classifier constructed from KDE classifies anomalies well, but its computational cost at query-time is potentially high, especially on large training sets. We combined different strengths from KDE and GP by using GP to learn this density. The resulting model can classify as well as OCKDE. Its computational cost at query-time is also reduced relative to KDE, not scaling with the size of training data. However, the classifier suffers from the weaknesses of computationally expensive global search at training time and tuning many parameters.

In this paper, two recently introduced local search heuristic algorithms [2, 3], LAHC and SCHC, are now introduced as simpler alternative search methods for GP. These simple systems perform remarkably well compared to a simple hill-climber. In comparison to a full GP system, they are straightforward, and depend only single parameter, *history length*. These new GP algorithms will be used as symbolic regression techniques for learning the density function. Thus, the proposed models will inherit the advantages of local search heuristic algorithms, few parameters and simplicity, and the OCC ability of KDE.

2 LAHC-GP AND SCHC-GP

LAHC-GP presented in Algorithm 1 is an iterative process, which starts with an initial GP individual s, and an initial list of L fitness values $\hat{F}_k, k \in \{0...L-1\}$. At each iteration, a new candidate s^* is created from the current one s through sub-tree mutation. If its fitness values $F(s^*)$ is not worse than that of the individual from L steps previously $\hat{F}_v$ or that of the current one $F(s)$, s^* will be accepted as the new current s. The slot $\hat{F}_v$ is then updated with the current fitness. It will stop after a fixed number of iterations.

Algorithm 1 Late Acceptance Hill-Climbing Genetic Programming

```
Produce an initial individual s
Calculate an initial fitness function F(s)
Specify L
for all k ∈ {0...L − 1} do F̂_k ← F(s)
Initial iteration I ← 0
while a stopping condition is NOT met do
    s* ← Mutation(s)
    Calculate fitness function F(s*)
    v ← I  mod L
    if F(s*) ≤ F̂_v ∨ F(s*) ≤ F(s) then accept candidate (s ← s*)
    else reject candidate (s ← s)
    Insert fitness value into the list F̂_v ← F(s)
    Increment the iteration I ← I + 1
```

SCHC-GP described in Algorithm 2 is very similar to LAHC-GP. However, the algorithm uses a fitness bound B_c instead of a list, and a counter n_c. At each iteration, if a candidate's fitness value is better than B_c or not worse than the current fitness, the candidate will be accepted as the new current one, the counter will be increased. B_c will be updated with the current fitness value if the counter exceeds L_c steps. It will stop after a fixed number of iterations.

GECCO '17 Companion, Berlin, Germany
 978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3076091

Algorithm 2 Step Counting Hill-Climbing Genetic Programming

1: Produce an initial individual s
2: Calculate an initial fitness function $F(s)$
3: Initial fitness bound $B_c \leftarrow F(s)$
4: Initial counter $n_c \leftarrow 0$
5: Specify L_c
6: **while** a stopping condition is NOT met **do**
7: $s^* \leftarrow Mutation(s)$
8: Calculate fitness function $F(s^*)$
9: **if** $F(s^*) < B_c \vee F(s^*) \leq F(s)$ **then** accept candidate $(s \leftarrow s^*)$
10: Increment the counter $n_c \leftarrow n_c + 1$
11: **else** reject candidate $(s \leftarrow s)$
12: **if** $n_c \geq L_c$ **then**
13: update the bound $B_c \leftarrow F(s)$
14: reset the counter $n_c \leftarrow 0$

Table 1: The performance of the five classifiers

Method	C-heart	ACA[1]	WBC	WDBC	R2L
OCSVM	0.759	0.820	0.991	0.950	0.859
OCKDE	0.773	0.835	0.991	0.953	0.900
OCGP	0.800	0.833	0.993	0.948	0.881
LAHC-GP	0.788	0.823	0.991	0.949	0.899
SCHC-GP	0.789	0.828	0.991	0.949	0.897

3 EXPERIMENTS

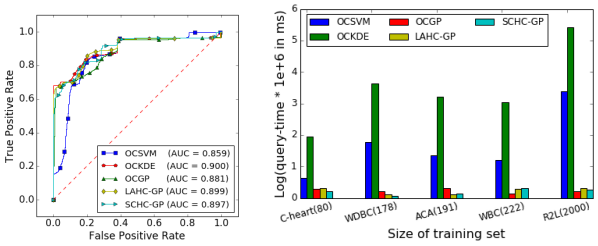
Our experiments are to compare LAHC-GP and SCHC-GP with OCGP, OCSVM, and OCKDE. Therefore, we will reproduce the experimental results as reported in [5]. We will also employ the same datasets, the same parameter settings of KDE and one-class SVM as in [5] for these experiments.

The first experiment is to tune the *history length* L and L_c based on the training errors[2]. Thus, we choose $L = 300$, $L_c = 200$. *Fitness evaluation budget* (the number of iterations) is equal to *population size* $\times$ *number of generation* of OCGP[3], 200000. The performance of the proposed models are evaluated on the datasets under two measurements, the Area Under ROC Curve (AUC) and query-time. The AUC values of the five classifiers are shown in Table 1 and Fig 1a. The average query-times[4] are plotted against the size of training sets shown in Fig 1b, and aspects of the computational cost at the query-time are also reported in Table 2.

4 DISCUSSION RESULTS AND CONCLUSIONS

Table 1 shows that LAHC-GP and SCHC-GP classifiers perform as well as OCGP and OCKDE, and often better than OCSVM in terms of classification accuracy. The proposed classifiers' accuracies approach closer to that of OCKDE than OCGP on some datasets. In Fig 1b, the query-time of LAHC-GP, SCHC-GP classifiers and OCGP are quite similar, and much lower than those of OCKDE and OCSVM. The query-time of OCKDE and OCSVM tend to increase when the size of training set increases, whereas those of three

[1] Australian Credit Approval [2] Tuning on five different values of *history length* ranging from 100 to 500. [3] This is because there is only one individual during the iteration. [4] The average query-time per example is calculated over 100 repetitions.



(a) The ROC curves **(b) The average query-times**

Figure 1: The performances of the five classifiers

Table 2: Aspects of the computational cost at the query-time

Dataset	Training Points	Support Vectors	GP Notes	LAHC-GP Notes	SCHC-GP Notes
C-heart	80	53	237.9	198.4	188.3
ACA	191	112	218.4	161.7	168.4
WBC	222	115	207.8	196.6	206.7
WDBC	178	121	182.0	161.6	150.5
R2L	2000	1001	197.2	207.7	196.9

others seem to be stable, and this is expected since the query time of a GP model is proportional to the number of nodes. This is also demonstrated by aspects of the computational cost at query-time reported in Table 2.

The results suggest that the performance of the proposed models is competitive with that of the full GP based OCC, hence the classification accuracy approaches or equals that of OCKDE, and often out-performs one-class SVM. The query-times of the proposed models are much less than those of OCSVM and OCKDE, and seem to not scale with the size of training data.

Overall, the proposed models not only inherit the ability of learning density from the full GP, but are also straightforward and less vulnerable to inadequate parameterization. These strengths mean that LAHC-GP and SCHC-GP are useful techniques for anomaly detection problems. The work of investigating *history length* and carrying out statistical test on the results is postponed to future research.

REFERENCES

[1] Charu C Aggarwal. 2015. Outlier analysis. In *Data Mining*. Springer, 237–263.
[2] Edmund K Burke and Yuri Bykov. 2017. The late acceptance Hill-Climbing heuristic. *European Journal of Operational Research* 258, 1 (2017), 70–78.
[3] Yuri Bykov and Sanja Petrovic. 2016. A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling. *Journal of Scheduling* 19, 4 (2016), 479–492.
[4] Van Loi Cao, Miguel Nicolau, and James McDermott. 2016. A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection. In *International Conference on Parallel Problem Solving from Nature*. Springer, 717–726.
[5] Van Loi Cao, Miguel Nicolau, and James McDermott. 2016. One-Class Classification for Anomaly Detection with Kernel Density Estimation and Genetic Programming. In *European Conference on Genetic Programming*. Springer, 3–18.
[6] Robert Curry and Malcolm I Heywood. 2009. One-class genetic programming. In *European Conference on Genetic Programming*. Springer, 1–12.
[7] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition* 58 (2016), 121–134.