One-Class Classification for Anomaly Detection with Kernel Density Estimation and Genetic Programming

Van Loi Cao(✉), Miguel Nicolau, and James McDermott

Natural Computing Research and Application Group,
University College Dublin, Dublin, Ireland
loi.cao@ucdconnect.ie, {miguel.nicolau,james.mcdermott2}@ucd.ie
http://ncra.ucd.ie

Abstract. A novel approach is proposed for fast anomaly detection by one-class classification. Standard kernel density estimation is first used to obtain an estimate of the input probability density function, based on the one-class input data. This can be used for anomaly detection: query points are classed as anomalies if their density is below some threshold. The disadvantage is that kernel density estimation is lazy, that is the bulk of the computation is performed at query time. For large datasets it can be slow. Therefore it is proposed to approximate the density function using genetic programming symbolic regression, before imposing the threshold. The runtime of the resulting genetic programming trees does not depend on the size of the training data. The method is tested on datasets including in the domain of network security. Results show that the genetic programming approximation is generally very good, and hence classification accuracy approaches or equals that when using kernel density estimation to carry out one-class classification directly. Results are also generally superior to another standard approach, one-class support vector machines.

Keywords: Anomaly detection · One-class classification · Kernel density estimation

1 Introduction

Anomaly detection is the problem of detecting samples or patterns in data that are different from expected behavior [3]. The nonconforming patterns are referred to a variety of names in different application domains, but the terms *anomalies* and *outliers* are common. Anomaly detection is applied in many fields such as intrusion detection, credit card fraud detection, insurance, health care, fault detection in safety critical systems. Anomaly detection plays an important role in a variety of application domains because anomalies in data often translate to critical, actionable information or potentially dangerous situations [1,3].

In network security, anomaly detection means the discrimination of illegal and malicious activities from normal connections or expected behavior of systems

M. Heywood et al. (Eds.): EuroGP 2016, LNCS 9594, pp. 3–18, 2016.
DOI: 10.1007/978-3-319-30668-1_1

[13,14]. The role of automated anomaly detection has become increasingly important in network security due to the widespread use of computer networks in recent years [13]. However, there are some issues that make constructing anomaly detection models challenging. One of the major issues is that anomalies are continuously evolving over time. The model built from current data may not be able to capture attacks or unauthorized accesses in the future [6]. Collecting the anomalous data is extremely difficult due to the privacy and security concerns of computer networks and the shortage of intrusive network traffic in host logs and events. Moreover, labeling such data is also a challenging and time-consuming task for experts in the domain and has potential problems [24]. Thus, in many situations only the normal class is available.

Because of these issues, one-class learning or novelty detection is a common method for anomaly detection. In *one-class classification* (OCC), only one class (the target) is available for constructing a classifier. The classifier is then used to distinguish whether a test instance belongs to the target class or the non-target class [29]. In this work, we use the terms target and non-target to refer to normal and anomaly respectively. More details about one-class learning and recent one-class methods are discussed in Sect. 2.

In this paper, we propose a one-class learning method by combining Genetic Programming (GP) with Kernel Density Estimation (KDE) (described in detail in Sect. 3.2). KDE has the ability to directly estimate density from data. It can thus be used as a one-class classifier by imposing a density threshold: points in low-density areas are classed as anomalies. However the computational cost of KDE at query time is potentially high, scaling with the number of training points. Fortunately, GP has the ability to approximate density. The result is a model where the computational cost at query time depends on the number of nodes in the GP tree. Therefore, in our system KDE is first used to estimate the density from target examples, and GP is then employed to approximate this density. The resulting one-class classifier not only yields high accuracy in detecting anomalies, but has reduced computational cost relative to KDE. Another potential advantage, not pursued in this paper, is that the resulting models are interpretable GP trees. The method is described in detail in Sect. 4.

The rest of this paper is organized as follows. In the next section, we briefly review some work related to one-class classification. In Sect. 3, we give a short introduction to GP for classification, and KDE. This is followed by a section proposing OCC using KDE and GP. Experiments, and Results and Discussion are presented in Sects. 5 and 6 respectively. The paper concludes with highlights and future directions.

2 Related Work

The concept of one-class classification was originated by Moya et al. [18]. One-class classification has rapidly emerged in a variety of fields from document classification [17], concept learning [9], novelty detection [2] to anomaly detection [8,20]. Based on the availability of training data, one-class classification can be categorized into three groups [11]: (1) learning from only target examples; (2) learning from target examples and unlabeled data; (3) learning from target examples and a small number of non-target examples. In terms of anomaly detection, we concentrate on (1), that is the one-class learning techniques that construct a model when non-target data is absent.

Tax and Duin [28,29] proposed a method called Support Vector Data Description (SVDD) to solve the problem of OCC. In the method, a hypersphere with minimum radius around the target examples in feature space is found, which encompasses almost all target instances. In order to achieve a sufficient decrease in the volume of the hypersphere, it possibly rejects some fraction when training this model. This illustrates a theme present in all one-class classification research, the trade-off between false positive and false negative rates. Tax [27] introduced different kernel functions to SVDD that make the method more flexible, and the Gaussian kernel was found to be the most suitable for many datasets. However, this technique requires a large number of target examples, and becomes inefficient in high dimension [11].

Instead of finding a hypersphere, Schölkopf [21,22] presented an alternative approach called one-class SVM. The one-class classifier is achieved by searching for a hyperplane with a maximum margin between the region containing target data and the origin in feature space. The target data is again mapped to feature space via a kernel. The efficacy of the method is evaluated on a handwritten digit dataset, and the results show that the classifier performs well.

Recently, several evolutionary algorithm approaches have been proposed for one-class classification problem [4,5,31]. Heywood and Curry have a long line of research, e.g [4,5] on multi-objective one-class genetic programming. In order to establish a more precise boundary, a large number of artificial data is generated to construct a two-class classifier. They combine a multi-objective fitness function with a local membership function to improve the search for specific regions of the target distribution. Its performance was compared to one-class v-SVM, bottleneck neural network (BNN) and two-class SVM, and the results show that the one-class GP performs consistently overall.

Cuong To [31] proposed a different approach to one-class problem by calculating the sum of the Euclidean distance from a target point to all target examples with an assumption that the distribution of the summed distance is normal. This is similar to an inversion of a kernel method since a kernel is a similarity whereas a Euclidean distance is a dissimilarity. They then used GP to approximate this sum. They then used two thresholds at 2.5 % and 97.5 % on training dataset to classify new examples. New points whose sum of distances was below the 2.5 percentile or above the 97.5 percentile of the data were classed as anomalies. Effectively, the lower threshold meant that points with extremely low sums of distances (i.e. directly in the middle of the target class) were wrongly classed as anomalies. It is possible that malfunctions in their GP approximation made this necessary. Our method is similar but avoids this error by using a single, upper threshold. We also gain flexibility by using a kernel instead of Euclidean distance.

In this work, we present a new approach for one-class classification problem. This is to combine Kernel Density Estimator and Genetic Programming to take advantage of their different strengths. While KDE can estimate the density directly from data with unknown distribution, GP has the ability to search for a function mapping from data to its density. Thus, the classifier not only produces accuracy as good as KDE, but also reduces time consuming on testing stage.

3 Preliminaries

In this section we briefly introduce Genetic Programming and its application in classification (Sect. 3.1), and introduce Kernel Density Estimation and its application in one-class classification (Sect. 3.2).

3.1 Genetic Programming

GP was popularized by Koza in the 1990s [12]. It is an evolutionary paradigm that is inspired by biological evolution. It aims to find good solutions to a diverse spectrum of problems in the form of computer programs. GP has the ability to represent solutions to many problems since the representation is highly flexible.

In particular, GP methods can be adapted to classification problems [16]. For two-class classification problems, a typical approach is to evolve real-valued GP trees, and then translate the numeric (real) values returned by them into class labels using a threshold. Typically, the threshold is zero. Similar methods can be applied for one-class classification (OCC): artificial data acts as the non-target class [4,5]. However, in this paper we propose a different GP approach to OCC. This is to use GP to approximate the density of the training set (and artificial data) originally given by KDE.

3.2 Kernel Density Estimation

Kernel density estimation is one of the most attractive non-parametric methods in the statistical literature for estimating a probability density function from a sample of points [32]. It estimates the density function directly from data with no assumption about the underlying distribution. It produces asymptotic convergence to any density function [23]. These advantages make KDE a general approach for many problems that do not assume any specific distribution of the density function.

Let $x_1, x_2, ..., x_n$ be a set of d-dimensional samples in $\mathbb{R}^d$ drawn from an unknown distribution with density function $p(x)$. An estimate $\hat{p}(x)$ of the density at x can be calculated using

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h \left(x - x_i \right) \tag{1}$$

where $K_h : \mathbb{R}^d \to \mathbb{R}$ is a kernel function with a parameter h called the *bandwidth*.

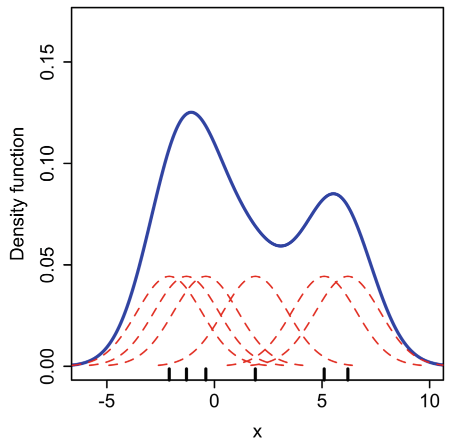


Fig. 1. The density distribution estimated by KDE. Figure reproduced from [34]

Two factors, kernel function and bandwidth play vital roles in KDE. A variety of kernel functions with different properties are typically chosen for KDE; eg. Gaussian, Uniform, Epanechnikov, Exponential, Linear and Cosine. The Gaussian kernel (Eq. 2) is probably the most common in applications and is the only one used in this paper. As illustrated in Fig. 1, in KDE each point contributes a small "bump" to the overall density, with its shape controlled by the kernel and bandwidth. The bandwidth parameter h controls the trade-off between bias of the estimator and its variance. This means that a large bandwidth leads to a very smooth (i.e. high-bias) density distribution while the density distribution is less smooth (i.e. high-variance) with a smaller bandwidth.

$$K_h\,(x) = \exp\,(-\frac{x^2}{2h^2}) \tag{2}$$

In terms of OCC, we construct a model from only the target class, and we typically do not assume any particular parametric distribution (e.g. Normal or uniform) on the target class. Thus, KDE is a suitable approach for estimating the density distribution of the target class. This can then be used to define a classifier by imposing a threshold in terms of density: new (query) points are classified as anomalies if they have a density lower than (say) 95 % of the training set. The choice of threshold in practice depends on the relative frequency of anomalies in the domain and the relative costs of false positives and false negatives.

A main drawback of KDE is its computational cost. KDE "remembers" all training data in order to compute the density of each new point. Thus, the larger the training data sample size, the greater the computational cost of querying new points. The computational cost of KDE can be markedly improved using space-partitioning trees, but the extent of the improvement depends on the bandwidth: for large values of bandwidth (which appear to give the best results in our experiments, described in Sect. 5), space-partitioning trees cannot eliminate as many points, hence the improvement is less. In any case, the complexity still depends on the size of the training data [7].

4 Proposed Approach

In this paper we use GP to approximate the function learned by KDE, in order to speed up the querying stage and remove the dependence on the size of the training data. A second potential benefit of this approach is that the density is expressed as a potentially interpretable symbolic expression, although we do not use this interpretation in this paper. Therefore, the GP classifier not only inherits the advantages of KDE, but also can reduce the computational cost of the querying stage.

However, if training data is drawn from a small area of the feature space, the classifier will lack the ability to predict density on points outside that area. One solution is to generate artificial data in the low-density regions of the space. Ideally, generating new data near the "border" between high and low-density regions will particularly improve the approximation in important areas. This will strengthen the classifier's discrimination on both target and non-target examples. More details of the proposed one-class genetic programming technique is described in the next subsection.

4.1 Description of Method

Let $T = \{x_1, x_2, ..., x_m\}$ be the target training set, where $x_i \in \mathbb{R}^n$ is a target instance and m is the number of samples in the target training set. We assume that T has been standardized, that is shifted and scaled to give zero mean and unit variance.

1. Kernel density estimation is employed to estimate the density distribution of the target training set. Using the kernel density estimator, we compute the density $d(x_i)$ for every target sample $x_i \in T$.
2. An artificial data set $A = \{x_{m+1}, x_{m+2}, ..., x_{m+q}\}$, is generated as described in Sect. 4.2. The KDE is used to compute the density $d(x_i)$ for each artificial example $x_i \in A$ against the target density distribution, where $i \in \{m+1, m+2, ..., m+q\}$.
3. Let $TS = T \cup A = \{x_1, x_2, ..., x_m, x_{m+1}, ..., x_{m+q}\}$, and $d(x_i)$ is the density of each point $x_i \in TS$, where $i \in \{1, 2, ..., m, m+1, ..., m+q\}$. We use GP symbolic regression with a root mean square error to search for a function f that satisfies the criterion in Eq. 3.

$$f\left(x_i\right) \approx d(x_i) \tag{3}$$

Therefore, the fitness function is given by

$$Fitness = \sqrt{\frac{\left(\sum_{i=1}^{m+p}\left(f(x_i) - d(x_i)\right)^2\right)}{(m+p)}} \tag{4}$$

where m and p are the numbers of the target training set and the artificial data set respectively, $d(x_i)$ is the density of sample $x_i \in TS$.

Now we have a function f that, like KDE, can predict density on query data but with less computational cost. To construct a classifier, we impose a threshold on f, just as in the case of KDE, for example at a level that will classify the 5 % of the training data with the lowest density values as anomalies. The choice of threshold depends on the domain and dataset.

4.2 Generating Artificial Data

The aim is to generate artificial data around the target training set and in the low-density regions of the target training set. As mentioned in Subsect. 4.1, our input data is standardized hence it is centered at the origin. However, it is infeasible when we only use standard methods (e.g. Gaussian, Uniform) to generate data in high dimensional feature spaces [4,5,30]. The probability of an artificial point being in or around the boundary of the target distribution is very small. Therefore, we propose a method to generate artificial data with more points in and around the target distribution.

There are two main steps to generate the artificial data in our method. We generate data in a hypersphere centered at the origin, uniformly in terms of radius. We then sample only points that are around the target training set and in the low-density regions of the target training set. More details of these steps are described below:

1. Generate uniformly in terms of radius
   (a) Data X is generated from Gaussian distribution in n dimensions with zero mean and unit standard deviation, see Fig. 2(a)

$$X \sim \mathcal{N}(0, 1) \tag{5}$$

   (b) Relocate all samples in X to the surface of the unit hypersphere, see Fig. 2(b). The direction of each point in X does not change, thus $X^{'}$ is uniformly distributed on the surface of the unit hypersphere:

$$X^{'} = \frac{X}{\|X\|} \tag{6}$$

   where $\|X\|$ is the Euclidean distance from each sample in X to the origin.

   (c) Uniformly generate U in one dimension with a range $[0, R]$

$$U \sim \mathcal{U}(0, R) \tag{7}$$

   where R is the maximum radius of a hypersphere. R is chosen to give the desired distribution, including points in low-density regions. The value for each dataset is given in the next section.

   (d) Rescale all objects in $X^{'}$ with a factor U:

$$X^{''} = U.X^{'} \tag{8}$$

   Now $X^{''}$ are uniform in terms of radius to the origin in the hypersphere with maximum radius R, see Fig. 2(c).

2. Sample data in and around the target training set
   (a) The density of each point in $X^{''}$ is computed against the target training set by KDE.
   (b) Any point in $X^{''}$ whose density is greater than a density value that is determined by a threshold $t_{overlap}$ is discarded. Threshold $t_{overlap}$ determines what percentage of the target training set is overlapped by the artificial data in terms of density. This is because in high-density regions we already have target examples.
   (c) Randomly sample a data set T from $X^{''}$ with a proportion p, where p is proportional to the number of examples in target training set.

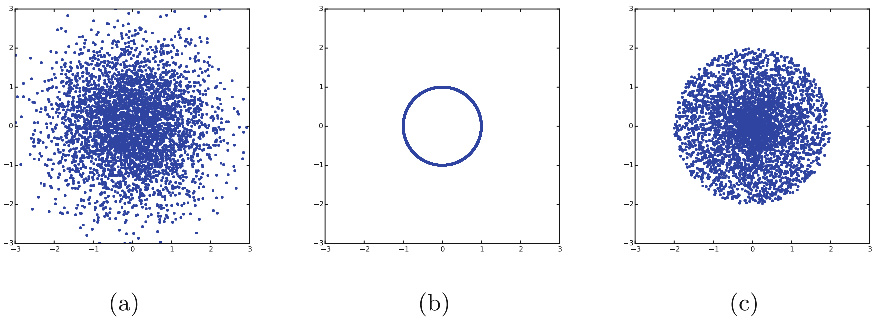


Fig. 2. Generate artificial data uniformly in terms of radius. (a) Gaussian distribution. (b) Uniform distribution on the surface of the unit hypersphere. (c) Distribution in an arbitrary hypersphere, uniform in terms of radius.

5 Experiments

5.1 Datasets

The goal of the experiments is to evaluate our method on one-class datasets. Thus, we choose datasets that have one class considered as target class and other classes treated as a non-target class [4,31]. Four datasets in UCI Machine Learning Repository [15] and NSL-KDD dataset [25] are employed for our experiments. In Wisconsin Breast Cancer Database (WBC), each instance contains 9 real-value attributes, one id and one class attribute that divides data into two classes, benign (458 instances) and malignant (241 instances). We remove

16 examples that contain missing data. There are 375 benign examples and 212 malignant examples in Wisconsin Diagnostic Breast Cancer (WDBC). Each instance is presented by 32 attributes (id, class attribute and 30 real-valued attributes). The third dataset is Cleveland heart disease (C-heart) that contains 164 examples for heart disease level 0, and 139 examples for heart disease level 1–4. There are 14 attributes (13 real-valued attributes and one class attribute). The last UCI dataset used in our experiments is the Australian Credit Approval dataset. There are 690 instances and each belongs to approval class or risk class (383 approval, 307 risk instances respectively). Each instance is described by 14 real-valued attributes and one class attribute.

For the four datasets, we randomly sample 50 percent of the target class for the target training set and the other 50 percent for the target testing set. However, all examples from non-target class is used for the non-target testing set. More details are presented in Table 1.

NSL-KDD dataset [26] is a filtered version of the KDD Cup 1999 dataset [10], which is in the domain of network security, after removing all redundant instances and making the task more difficult. In NSL-KDD, a connection is represented by 41 attributes (38 numeric continuous and discrete, and 3 categorical attributes). Each record is labeled as either normal or as a specific kind of attack belonging to one of the four main categories: Denial of Service (DoS), Remote to Local (R2L), User to Local (U2R) and Probe. NSL-KDD consists of two datasets: $KDDTrain^+$ and $KDDTest^+$ which are drawn from different distributions.

In this work, we plan to conduct our experiments on the R2L attack group. This is because the aim of the search is to reduce computational cost, thus it is efficient to reduce redundant features [33]. Moreover, the records from the R2L group are slightly similar to normal connections due to the fact that they are based on some content features of network traffic. This makes them more difficult to classify than DoS or Probe attack groups [13,14,24]. Based on a previous feature selection research [33], we choose only a subset of 10 features from 41 features in NSL-KDD for detecting the R2L attack category in our experiment. Several of the variables chosen are categorical or discrete. We simply treat them as real-valued (e.g. we map *service* values TCP, UDP and ICMP to values 1, 2 and 3). As shown in Sect. 6 this gives good results.

In our experiments, we randomly sample 2000 normal instances from $KDDTrain^+$ for the target training set, whereas 2000 normal examples and 2000 R2L examples are randomly selected from $KDDTest^+$ for the target testing set and the non-target testing set respectively. More details are presented in Table 1.

We use the proposed method in Sect. 4.2 to generate artificial data for our experiments. The threshold $t_{overlap}$ for the artificial data was set so that 10 % of the target training set was below it in terms of density. Figure 3 shows the density distribution of the target training set and artificial data with 10 % overlap. The size of the artificial training set was chosen to be approximately double the size of the target training set, except for datasets where this was small (< 100) or large (> 1000). Too large a number could lead to a good GP approximation in the low-density region and a poor GP approximation in the high-density region, while too small a number would lead to the opposite. The numbers of examples in target training set, artificial dataset, testing sets are shown in Table 1.

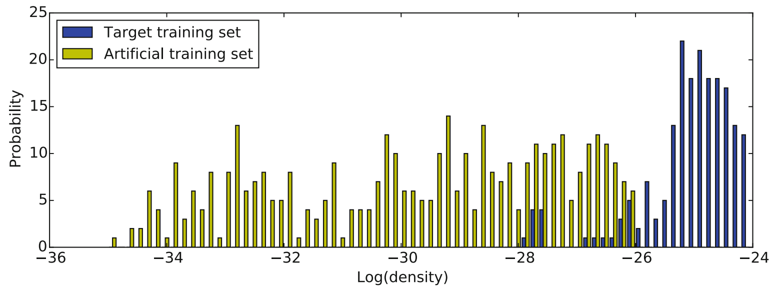


Fig. 3. The density distribution of target training set and artificial data

Table 1. One-class classification datasets

Dataset	Features	Training set		Testing set	
		Target	Artificial	Target	Non-target
C-heart	13	80	350	80	137
Australia credit card	14	191	400	192	307
WBC	9	222	500	222	239
WDBC	30	178	400	179	212
R2L (NSL-KDD)	10	2000	1000	2000	2000

5.2 Experimental Settings

One preliminary experiment and one main experiment on GP, KDE and one-class SVM are conducted in order to evaluate the proposed one-class GP on accuracy and runtime of the resulting models. The five datasets in Table 1 are used for the experiments. We use the terms OCGP, OCKDE, and OCSVM to refer to one-class GP, KDE, and SVM classifiers, respectively. The choice of threshold for classifier in practice varies from domain to domain, but in our experiments we evaluate AUC using many values of the threshold (and we do the same for all methods, OCGP, OCKDE, OCSVM). More details of the two experiments are described below:

In the preliminary experiment we investigated the effect of the bandwidth parameter. We chose the Gaussian kernel and used cross-validation to choose its bandwidth value for KDE. That is, the accuracy of the density estimation of the input data (not the accuracy on a one-class classification task) was used to choose an optimal value for bandwidth. Accuracy was measured as mean integrated squared error (MISE). The resulting bandwidth was in all cases low (about 0.8). When used in a one-class classification task, this bandwidth achieved poor performance. Similar results were found for OCSVM. Therefore, for remaining experiments we chose a larger bandwidth value (2.0) that gave worse results on density estimation, but better results on the one-class classification task.

The main experiment first plans to investigate KDE in terms of one-class classification. This is a basic framework to evaluate OCGP. We set up the same kernel function and bandwidth as described in the preliminary experiment. Kernel density estimator is run on five datasets in Table 1, and we calculate the AUC values for OCKDE on every dataset.

Secondly, the experiment aims to examine how efficiently OCGP performs on these datasets. Evolutionary parameters and the parameters for generating artificial data are presented in Table 2. We calculate the mean of AUC values over 50 runs to evaluate OCGP, and select the individual with median AUC value over 50 runs to draw ROC curves against OCKDE and OCSVM. The computational cost of OCGP on testing stage is calculated as the mean of the numbers of nodes in the best-of-run GP trees over 50 runs.

Finally, the experiment will examine one-class SVM [22] in order to compare its performance to OCGP. We set up the same kernel function and bandwidth as KDE, and $\nu = 0.5$. We use one-class SVM from sklearn [19] to run experiments over the five datasets. The AUC values and the number of support vector are computed. The results from the three experiments on the five datasets in Table 1 are presented in Tables 3 and 4, and Fig. 4.

Table 2. Parameter settings

Generate artificial data	
R	10
$t_{overlap}$	10 percent
KDE and OC SVM parameters	
Bandwidth	2.0
Kernel function	Gaussian
GP parameters	
Population size	400
Number of generation	500
Crossover probability	0.9
Mutation probability	0.1
Selection	Tournament
Tournament size	3
Function set	$\{+, -, \times, /, \exp, \mathrm{sqr}, \mathrm{sqrt}\}$

6 Results and Discussion

This section presents the experimental results of evaluating the proposed one-class GP classifier on the five datasets. The performance of the one-class GP classifier is evaluated along two measurements, Area Under ROC Curve (AUC) and computational cost on querying new instances. The results are shown in Tables 3 and 4, and in Fig. 4 the ROC curves of OCGP median over 50 runs on the five datasets are shown against those of OCSVM and OCKDE.

Table 3 illustrates the AUC values when applying the three one-class classifiers on the five datasets. It can be seen from the table that OCGP performs as well as OCKDE, and better than OCSVM in most of datasets in terms of accuracy.

The mean of AUC values from OCGP are close to OCKDE on C-heart, Australian Credit Card, WBC and R2L (see Table 3). In comparison to OCc, the AUC values from OCSVM are lower than those from OCGP on C-heart, Australian Credit Card and R2L. Conversely, OCGP does not produce a good accuracy on WDBC.

In general, these results suggest that the proposed OCGP tends to perform efficiently on the datasets that are low dimension (around 10 or 14 dimensions), but in high dimension (the WDBC dataset) the classifier may produce a poorer classification accuracy than OCKDE and OCSVM.

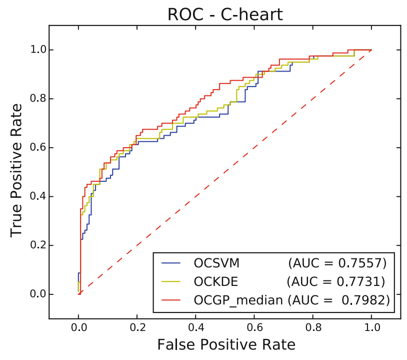
The ROC curves are demonstrated in Fig. 4. In this figure, we draw the ROC curve of OCGP$_{median}$ against the ROC curves of OCKDE and OCSVM. The term OCGP$_{median}$ refers to the GP individual that produces the median AUC

Table 3. The AUC results from three classifiers

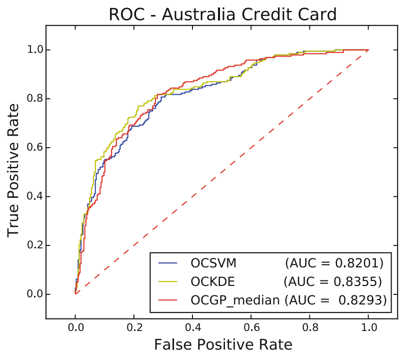
Dataset	AUC		
	OCKDE	OCSVM	$OCGP_{mean}$
C-heart	0.7731	0.7557	0.7874
Australian credit card	0.8355	0.8201	0.8286
WBC	0.9908	0.9907	0.9904
WDBC	0.9529	0.9500	0.9204
R2L	0.9001	0.8592	0.8727

Table 4. The computational cost for different techniques at query time

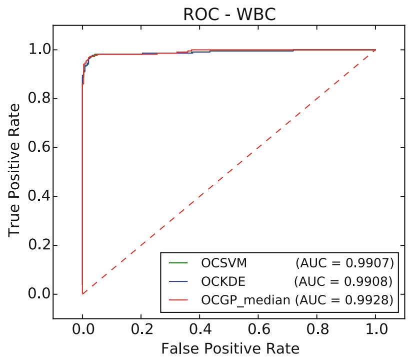
Dataset	Feature	Training points	Support vectors	GP nodes
C-heart	13	80	53	237.86
Australia credit card	14	191	112	218.36
WBC	9	222	115	207.8
WDBC	30	178	121	182
R2L	10	2000	1001	197.2



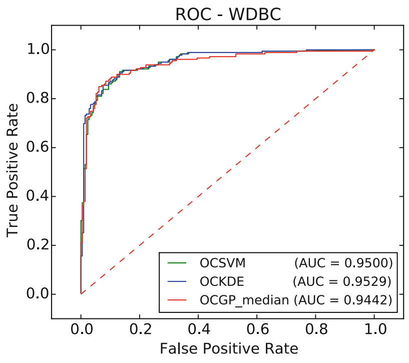
(a)



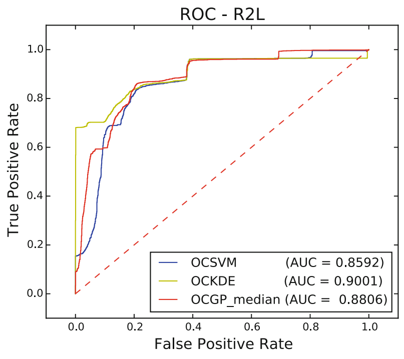
(b)



(c)



(d)



(e)

Fig. 4. The receiver operating characheristic from the three classifiers over the five datasets. (a) ROC on C-heart. (b) ROC on Australia credit card. (c) ROC on WBC (d) ROC on WDBC. (e) ROC on R2L

value over 50 runs. It can be seen from the figure that the ROC curve from OCGP$_{median}$ is higher than ROC curves from OCSVM in four datasets again with the single exception of WDBC.

Table 4 shows the computational cost for the three techniques at query time. Overall, the computational cost of OCGP does not depend on the size and dimension of training set. For OCGP, the average number of nodes in GP tree is around 200 on all five datasets. Conversely, the number of support vectors of OCSVM tends to increase with the number of training points. The runtime of OCKDE depends on the number of training points. This result suggests that OCGP is the least computationally expensive method at query time.

However, the training time of OCGP is much slower than OCKDE and OCSVM. It is approximately 10 min for constructing OCGP in comparison to a few seconds for building OCKDE or OCSVM models.

Overall, the results in this section suggest that the one-class GP classifier has superior scaling of query runtime to OCKDE and OCSVM in performance at query time, and its accuracy is often similar to the accuracy of OCKDE and higher than the accuracy of OC SVM.

7 Conclusion and Further Work

This paper has presented a novel approach to one-class classification. It aims to retain the biggest advantage of KDE, that is accurate modeling of the training data, while avoiding the main disadvantage, that is slowness at query time, when the training dataset is large. It aims to achieve this by modeling the output of KDE using GP. This requires extra training time, but the time to query new points does not then depend on the size of the training dataset. The output of the resulting GP model can be regarded as a density value. The model can be used to carry out one-class classification by imposing a threshold on this density. This threshold can be varied to reflect the desired false positive/false negative balance in any particular application.

In order to make the GP approximation work, it is necessary to generate some artificial training samples in low-density regions of the space, since otherwise GP has no knowledge of these regions. We have found suitable methods of doing this for our datasets. The density values for these samples can be found directly from the already-trained KDE.

Results have shown that with these artificially-generated samples, GP succeeds in approximating the density function given by KDE. While the resulting one-class classification accuracy cannot be expected to exceed that of using KDE, it often approaches or equals it. It is also often superior to the accuracy achieved using another standard method, one-class SVM.

The one-class classification problem has many important applications. One example is in network security. In this domain, query points are streamed in at a high rate, hence the performance gain achieved using the GP approximation is a valuable contribution. We have shown that our method can provide good accuracy in this domain.

Our method still has some limitations. In particular, we wish to adapt the method of generating artificial data to work well in higher-dimensional spaces and to deal with binary or categorical variables. We will also investigate alternative methods of setting the bandwidth automatically. We will apply and tailor our method to other network security datasets. Finally, we will report on the measured improvement in runtime on large datasets, and compare our method to alternatives such as the Autoencoder.

Acknowledgements. This work is funded by Vietnam International Education Development (VIED) and by agreement with the Irish Universities Association.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer Science & Business Media, New York (2013)
2. Bishop, C.M.: Novelty detection and neural network validation. In: IEE Proceedings on Vision, Image and Signal Processing, vol. 141, pp. 217–222. IET (1994)
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009)
4. Curry, R., Heywood, M.: One-class learning with multi-objective genetic programming. In: ISIC 2007 IEEE International Conference onSystems, Man and Cybernetics, pp. 1938–1945. IEEE (2007)
5. Curry, R., Heywood, M.I.: One-class genetic programming. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 1–12. Springer, Heidelberg (2009)
6. Fiore, U., Palmieri, F., Castiglione, A., De Santis, A.: Network anomaly detection with the restricted Boltzmann machine. Neurocomputing **122**, 13–23 (2013)
7. Gray, A.G., Moore, A.W.: Nonparametric density estimation: toward computational tractability. In: SDM, pp. 203–211. SIAM (2003)
8. Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., Kanamori, T.: Statistical outlier detection using direct density ratio estimation. Knowl. Inf. Syst. **26**(2), 309–336 (2011)
9. Japkowicz, N.: Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification. Ph.D. thesis, Rutgers, The State University of New Jersey (1999)
10. KDD Cup Dataset (1999). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
11. Khan, S.S., Madden, M.G.: A survey of recent trends in one class classification. In: Coyle, L., Freyne, J. (eds.) AICS 2009. LNCS, vol. 6206, pp. 188–197. Springer, Heidelberg (2010)
12. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT press, Cambridge (1992)
13. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf. Syst. Secur. (TiSSEC) **3**(4), 227–261 (2000)
14. Lee, W., Stolfo, S.J., Mok, K.W.: A data mining framework for building intrusion detection models. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy, 1999, pp. 120–132. IEEE (1999)

15. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
16. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, vol. 2, pp. 1070–1077. IEEE (2001)
17. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**, 139–154 (2002)
18. Moya, M.M., Koch, M.W., Hostetler, L.D.: One-class classifier networks for target recognition applications. Technical report, Sandia National Labs., Albuquerque, NM (United States) (1993)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
20. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In: ICDM 2006, Sixth International Conference on Data Mining, pp. 488–498. IEEE (2006)
21. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J.: SV estimation of a distributions support. Adv. Neural Inf. Process. Syst. **12** (1999)
22. Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.C.: Support vector method for novelty detection. NIPS **12**, 582–588 (1999)
23. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. John Wiley & Sons, New York (2015)
24. Shafi, K., Abbass, H.A.: Evaluation of an adaptive genetic-based signature extraction system for network intrusion detection. Pattern Anal. Appl. **16**(4), 549–566 (2013)
25. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD cup 99 data set. In: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009 (2009)
26. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: NSL-KDD dataset (2012). http://www.iscx.ca/NSL-KDD
27. Tax, D.M.: One-class classification. Delft University of Technology (2001)
28. Tax, D.M., Duin, R.P.: Data domain description using support vectors. In: ESANN, vol. 99, pp. 251–256 (1999)
29. Tax, D.M., Duin, R.P.: Support vector domain description. Pattern Recogn. Lett. **20**(11), 1191–1199 (1999)
30. Tax, D.M., Duin, R.P.: Uniform object generation for optimizing one-class classifiers. J. Mach. Learn. Res. **2**, 155–173 (2002)
31. To, C., Elati, M.: A parallel genetic programming for single class classification. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1579–1586. ACM (2013)
32. Wand, M.P., Jones, M.C.: Kernel Smoothing. CRC Press, Boca Raton (1994)
33. Wang, W., Gombault, S., Guyet, T.: Towards fast detecting intrusions: using key attributes of network traffic. In: ICIMP 2008, The Third International Conference on Internet Monitoring and Protection, pp. 86–91. IEEE (2008)
34. Wikipedia: Kernel density estimation – Wikipedia, the free encyclopedia (2015). https://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&oldid=690734894