Isolation-Based Anomaly Detection

Article *in* ACM Transactions on Knowledge Discovery from Data · March 2012

DOI: 10.1145/2133360.2133363

3 authors, including:

Fei Tony Liu
Monash University (Australia)
17 PUBLICATIONS **4,955** CITATIONS
SEE PROFILE

Zhi-Hua Zhou
Nanjing University
539 PUBLICATIONS **54,586** CITATIONS
SEE PROFILE

Isolation-based Anomaly Detection

Fei Tony Liu and Kai Ming Ting
Gippsland School of Information Technology
Monash University
and
Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University

Anomalies are data points that are few and different. As a result of these properties, we show that, anomalies are susceptible to a mechanism called *isolation*. This paper proposes a method called Isolation Forest (*i*Forest) which detects anomalies purely based on the concept of isolation without employing any distance or density measure—fundamentally different from all existing methods.

As a result, *i*Forest is able to exploit subsampling (i) to achieve a low linear time-complexity and a small memory-requirement, and (ii) to deal with the effects of swamping and masking effectively. Our empirical evaluation shows that *i*Forest outperforms ORCA, one-class SVM, LOF and Random Forests in terms of AUC, processing time, and it is robust against masking and swamping effects. *i*Forest also works well in high dimensional problems containing a large number of irrelevant attributes, and when anomalies are not available in training sample.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Algorithm, Design, Experimentation

Additional Key Words and Phrases: Anomaly detection, outlier detection, ensemble methods, binary tree, random tree ensemble, isolation, isolation forest

1. INTRODUCTION

Anomalies are data patterns that have different data characteristics from normal instances. The ability to detect anomalies has significant relevance, and anomalies

Author's addresses:
F. T. Liu & K. M. Ting, Monash University, Gippsland Campus, Churchill, Victoria, 3842, Australia; email: {tony.liu},{kaiming.ting}@monash.edu.
Z.-H. Zhou, National Key Laboratory for Novel Software Technology, Nanjing University, 22 Hankou Road, Nanjing 210093, China; email:zhouzh@lamda.nju.edu.cn. This research was supported by the Australian Postgraduate Awards (APA) and the Information and Communications Technologies (ICT) Postgraduate Research Scholarships, the National Science Foundation of China (61073097, 61021062), the National Fundamental Research Program of China (2010CB327903) and the Jiangsu Science Foundation (BK2008018).

often provides critical and actionable information in various application domains. For example, anomalies in credit card transactions could signify fraudulent use of credit cards. An anomalous spot in an astronomy image could indicate the discovery of a new star. An unusual computer network traffic pattern could stand for an unauthorised access. These applications demand anomaly detection algorithms with high detection accuracy and fast execution.

Most existing anomaly detection approaches, including classification-based methods [Abe et al. 2006; Shi and Horvath 2006], Replicator Neural Network (RNN) [Williams et al. 2002], one-class SVM [Tax and Duin 2004] and clustering-based methods [He et al. 2003], construct a profile of normal instances, then identify anomalies as those that do not conform to the normal profile. Their anomaly detection abilities are usually a 'side-effect' or by-product of an algorithm originally designed for a purpose other than anomaly detection (such as classification or clustering). This leads to two major drawbacks: (i) these approaches are not optimized to detect anomalies—as a consequence, these approaches often under-perform resulting in too many false alarms (having normal instances identified as anomalies) or too few anomalies being detected; (ii) many existing methods are constrained to low dimensional data and small data size because of the legacy of their original algorithms.

This paper proposes a different approach that detects anomalies by isolating instances, without relying on any distance or density measure. To achieve this, our proposed method takes advantage of two quantitative properties of anomalies: i) they are the minority consisting of **few** instances, and ii) they have attribute-values that are very **different** from those of normal instances. In other words, anomalies are 'few and different', which make them more susceptible to a mechanism we called Isolation. Isolation can be implemented by any means that separates instances. We opt to use a binary tree structure called isolation tree (iTree), which can be constructed effectively to isolate instances. Because of the susceptibility to isolation, anomalies are more likely to be isolated closer to the root of an iTree; whereas normal points are more likely to be isolated at the deeper end of an iTree. This forms the basis of our method to detect anomalies. Although, this is a very simple mechanism, we show in this paper that it is both effective and efficient in detecting anomalies.

The proposed method, called Isolation Forest (iForest), builds an ensemble of iTrees for a given data set; anomalies are those instances which have short average path lengths on the iTrees. There are two training parameters and one evaluation parameter in this method: the training parameters are the number of trees to build and subsampling size; the evaluation parameter is the tree height limit during evaluation. We show that iForest's detection accuracy converges quickly with a very small number of trees; it only requires a small subsampling size to achieve high detection accuracy with high efficiency; and the different height limits are used to cater for anomaly clusters of different density.

Apart from the key difference in using isolation as the means to detect anomalies, iForest is distinguished from existing model-based (e.g. [Abe et al. 2006; He et al. 2003]), link-based (e.g. [Ghoting et al. 2004]), depth-based (e.g. [Rousseeuw and Leroy 1987]), distance-based (e.g. [Knorr and Ng 1998]) and density-based methods

(e.g. [Breunig et al. 2000]) in the follow ways:

I The characteristic of isolation trees enables them to exploit subsampling to an extent that is not feasible in existing methods (more details are provided in Section 5.5).

II *i*Forest utilizes no distance or density measures to detect anomalies. This eliminates a major computational cost of distance calculation in all distance-based and density-based methods.

III *i*Forest has a linear time complexity with a small constant and a minimal memory requirement; it is an algorithm with constant training time and space complexities. To the best of our knowledge, the best performing existing method achieves only approximate linear time and space complexities [Angiulli and Fassetti 2009].

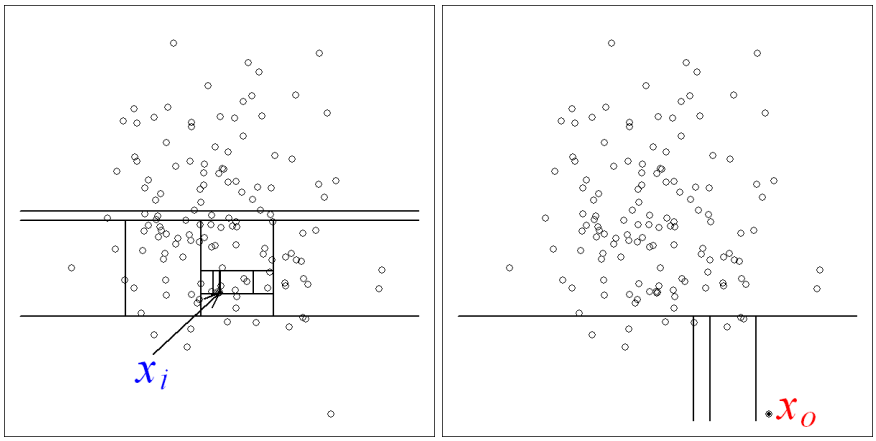
IV *i*Forest has the capacity to scale up to handle extremely large data size and high-dimensional problems with a large number of irrelevant attributes (Section 5.6).

This paper only concerns with unsupervised, non-parametric approaches for multivariate data anomaly detection and we focus on continuous-valued data only. We assume that all attributes in a data set contribute equally to the anomaly detection and we do not deal with conditional anomalies [Song et al. 2007] in this paper.

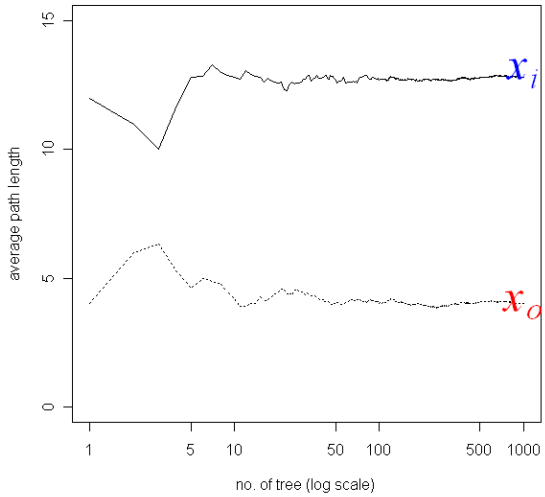
This paper is organised as follows: In Section 2, we demonstrate isolation at work using an *i*Tree that recursively partitions data. In Section 3, we compare isolation, density and distance measures to understand their fundamental differences. In Section 4, we provide the algorithms to construct *i*Trees and *i*Forest. A new anomaly score based on *i*Trees is also proposed. We utilize the subsampling size and evaluation height-limit to tackle the problems of swamping and masking. Section 5 empirically compares *i*Forest with four state-of-the-art anomaly detectors; we also analyse *i*Forest's detection performance (i) under different parameter settings, (ii) with increasing number of dense anomalies (masking effect), (iii) when the distance between anomalies and normal points reduces (swamping effect), (iv) high dimensionality in data, and (v) when anomalies are not available in training sample. Section 6 surveys the related work, Section 7 describes possible future work and Section 8 concludes this paper. Extending from the preliminary version of this article [Liu et al. 2008a], we have enriched Section 2 with extra illustrations and new materials can be found in Sections 3, 4.5, 5.2, 5.3, 5.4, 5.5, 5.7, 6, 7 and Appendices A, B, C, D and E.

2. ISOLATION AND ISOLATION TREES

In this paper, the term *isolation* means 'separating an instance from the rest of the instances'. In general, an isolation-based method measures individual instances' susceptibility to be isolated; and anomalies are those that have the highest susceptibility. To realize the idea of isolation, we turn to a data structure that naturally isolates data. In randomly generated binary trees where instances are recursively partitioned, these trees produce noticeable shorter paths for anomalies since (a) in the regions occupied by anomalies, less anomalies result in a smaller number of partitions – shorter paths in a tree structure, and (b) instances with distinguishable



(a) Isolating x_i (b) Isolating x_o



(c) Average path lengths converge

Fig. 1. Anomalies are more susceptible to isolation and hence have short path lengths. Given a Gaussian distribution (135 points), (**a**) a normal point x_i requires twelve random partitions to be isolated; (**b**) an anomaly x_o requires only four partitions to be isolated. (**c**) averaged path lengths of x_i and x_o converge when the number of trees increases.

attribute-values are more likely to be separated early in the partitioning process. Hence, when a forest of random trees collectively produce shorter path lengths for some particular points, they are highly likely to be anomalies.

In Figures 1(a) and 1(b), we observe that a normal point, x_i, generally requires more partitions to be isolated. The opposite is also true for an anomaly, x_o, which generally requires less partitions to be isolated. In this example, partitions are generated by randomly selecting an attribute and then randomly selecting a split value between the maximum and minimum values of the selected attribute. Since recur-

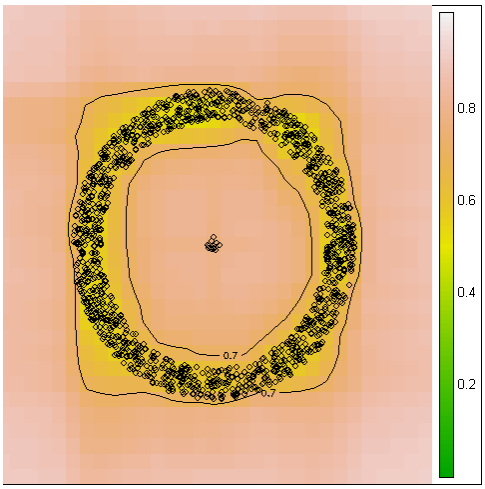


Fig. 2. *i*Forest is able to detect not only outlying scattered points, it can also detect anomalies surrounded by normal points as shown above. Thirteen anomalies are separated from surrounding normal points by high anomaly scores (> 0.7). Anomaly score ranges from 0 to 1 and it will be introduced in Section 4.3.

sive partitioning can be represented by a tree structure, the number of partitions required to isolate a point is equivalent to the traversal of path length from the root node to a terminating node. In this example, the path length of x_i is greater than the path length of x_o.

Since each partition is randomly generated, individual trees are generated with different sets of partitions. We average path lengths over a number of trees to find the expected path length. Figure 1(c) shows that the average path lengths of x_o and x_i converge when the number of trees increases. Using 1000 trees, the average path lengths of x_o and x_i converge to 4.0 and 12.8 respectively. It shows that anomalies have path lengths shorter than normal instances.

In addition to detecting scattered outliers as shown above, *i*Forest is also capable of detecting anomalies surrounded by normal points. Figure 2 illustrates such a scenario in which normal points form a ring shape and anomalies are at the 'centre' of the ring rather than on the outside. An anomaly score contour with two contour lines of 0.7 from an *i*Forest is shown in the figure. The anomalies have anomaly scores above 0.7 and normal points under 0.7. Anomaly score formulation will be detailed in Section 4.3.

Definition : **Isolation Tree.** Let T be a node of an isolation tree. T is either an external-node with no child, or an internal-node with one test and exactly two daughter nodes (T_l, T_r). A test at node T consists of an attribute q and a split value p such that the test $q < p$ determines the traversal of a data point to either T_l or T_r.

Let $X = \{x_1, ..., x_n\}$ be the given data set of a d-variate distribution. A sample of ψ instances $X' \subset X$ is used to build an isolation tree (*i*Tree). We recursively divide X' by randomly selecting an attribute q and a split value p, until either: (i)

the node has only one instance or (ii) all data at the node have the same values. An iTree is a *proper binary tree*, where each node in the tree has exactly zero or two daughter nodes. Assuming all instances are distinct, each instance is isolated to an external node when an iTree is fully grown, in which case the number of external nodes is ψ and the number of internal nodes is $\psi - 1$; the total number of nodes of an iTrees is $2\psi - 1$; and thus the memory requirement is bounded and only grows linearly with ψ. For the definitions of symbols and notations, please refer to Table I for details.

x	a data point
X	a data set of n instances
n	number of data points in a data set, $n = \|X\|$
m	index of data point x_m, $m \in \{0, ..., n-1\}$
Q	a set of attributes
d	number of attributes, $d = \|Q\|$
q	an attribute
T	a tree or a node
t	number of trees
$h(x)$	returns the path length of x
$hlim$	evaluation height limit
ψ	subsampling size
l	a possible path length
s	an anomaly score or function which returns an anomaly score
k	the number of nearest neighbours
a	contamination level of anomalies in a data set
cl	number of anomaly clusters

Table I. Symbols and Notations

The task of anomaly detection is to provide a ranking that reflects the degree of anomaly. Using iTrees, the way to detect anomalies is to sort data points according to their average path lengths; and anomalies are points that are ranked at the top of the list. We define path length as follows:

Definition : **Path Length** $h(x)$ of a point x is measured by the number of edges x traverses an iTree from the root node until the traversal is terminated at an external node.

We employ path length as a measure of the degree of susceptibility to isolation:

- short path length means high susceptibility to isolation,
- long path length means low susceptibility to isolation.

A probabilistic explanation of iTree can be found in Appendix A.

3. ISOLATION, DENSITY AND DISTANCE MEASURES

In this paper, we assert that path-length-based isolation is more appropriate for the task of anomaly detection than the basic density and distance measures.

Using basic density measures, the assumption is that *'Normal points occur in dense regions, while anomalies occur in sparse regions'*. Using basic distance measures, the basic assumption is that *'Normal point is close to its neighbours and anomaly is far from its neighbours'* [Chandola et al. 2009].

There are violations to these assumptions, e.g., high density and short distance do not always imply normal instances; likewise low density and long distance do not always imply anomalies. When density or distance is measured in a local context, which is often the case, points with high density or short distance could be anomalies in the global context of the entire data set. However, there is no ambiguity in path-length-based isolation and we demonstrate that in the following three paragraphs.

In density based anomaly detection, anomalies are defined to be data points in regions of low density. Density is commonly measured as (a) the reciprocal of the average distance to the k-nearest neighbours (the inverse distance) and (b) the count of points within a given fixed radius [Tan et al. 2005].

In distance based anomaly detection, anomalies are defined to be data points which are distant from all other points. Two common ways to define distance-based anomaly score are (i) the distance to k^{th} nearest neighbour and (ii) the average distance to k-nearest neighbours [Tan et al. 2005]. One of the weaknesses in these density and distance measures is their inability to handle data sets with regions of different densities[1]. Also, for these methods to detect dense anomaly clusters, k has to be larger than the size of the largest anomaly cluster. This creates a search problem: finding an appropriate k to use. Note that a large k increases the computation substantially.

On the surface, the function of an isolation measure is similar to a density measure or a distance measure, i.e., isolation ranks scattered outlying points higher than normal points. However, we find that path length based isolation behaves differently from a density or distance measure, under data with different distributions. In Figure 3, dense points on the left (anomalies) are at the fringe of a normal cluster on the right. The path length, density (k-nn) and k^{th}nn distance are plotted in Figures 3(a) and 3(b). We use $k = 10$ in both density and distance calculations. In Figure 3(a), density reports a high density for the anomaly points and low density for the normal cluster. In Figure 3(b), k^{th}nn distance reports a small distance for the anomaly points and a slightly larger distance for the normal cluster. These results based on distance and density measures are counter-intuitive. Path length, however is able to address this situation by giving the isolated dense points shorter path lengths. The main reason for this is that path length is grown in adaptive context, in which the context of each partitioning is different, from the first partition (the root node) in the context of the entire data set, to the last partition (the leaf node) in the context of local data-points. However, density (k-nn) and k^{th}nn distance only concern with k neighbours (local context) and fail to take the context of the entire data set into consideration.

In summary, we have compared three fundamental approaches to detect anomalies; they are isolation, density and distance. We find that the isolation measure

[1]Modified density-based methods, e.g., LOF [Breunig et al. 2000], are able to handle regions of different densities.

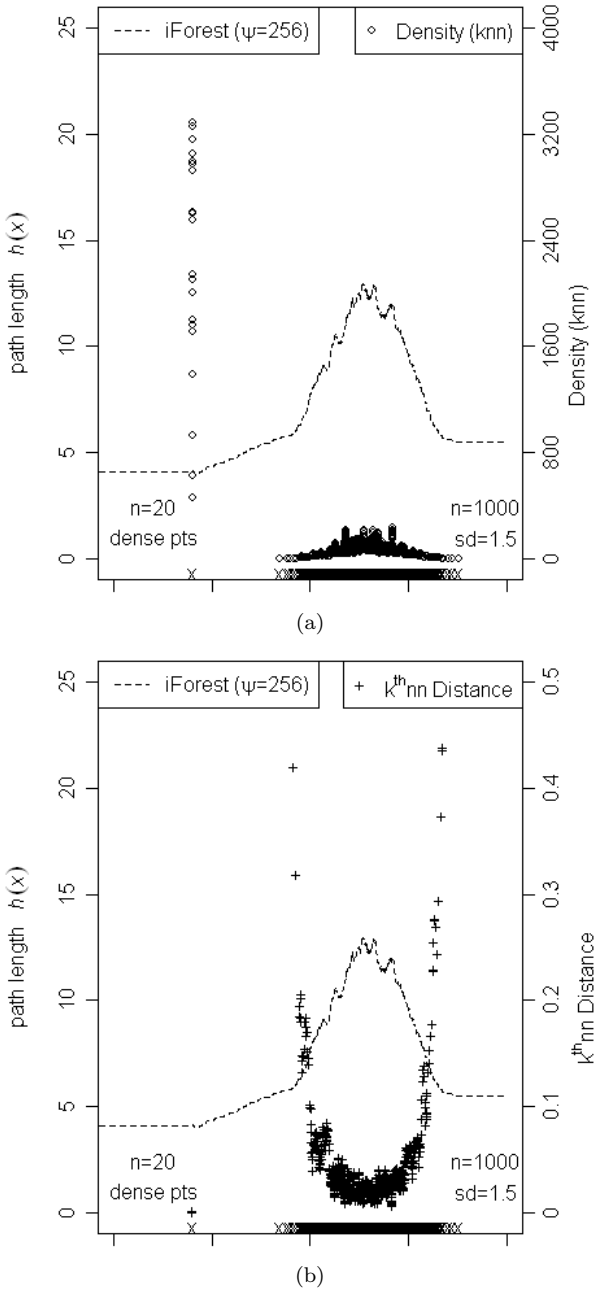


Fig. 3. This example show that points with high density and short distance (dense points on the left) could well be anomalies. On the other hand, low density points and points with long distance (the normal distribution) could also be normal instances. Note that path length is able to correctly detect the dense cluster as anomalies, by giving it shorter path length. Using a normal distribution of a dense cluster of twenty (on the left) and a thousand points (on the right), we compare path length with (a) density (k-nn) and (b) k^{th}nn distance, $k = 10$.

(path length) is able to detect both clustered and scattered anomalies; whereas both distance and density measures can only detect scattered anomalies. While there are many ways to enhance the basic distance and density measures, the isolation measure is better because no further 'adjustment' to the basic measure is required to detect both clustered and scattered anomalies.

4. ANOMALY DETECTION USING IFOREST

In this section, we describe the detail mechanism of *i*Forest and formulate an anomaly score that is meaningful for anomaly detection. Also, we explain why using small sub-samples brings about better isolation models and examine the changes of detection behaviour by adjusting the evaluation height-limit.

Anomaly detection using *i*Forest is a two-stage process. The first (training) stage builds isolation trees using sub-samples of the given training set. The second (evaluation) stage passes test instances through isolation trees to obtain an anomaly score for each instance.

4.1 Training Stage

In the training stage, *i*Trees are constructed by recursively partitioning a sub-sample X' until all instances are isolated. Details of the training stage can be found in Algorithms 1 and 2. Each *i*Tree is constructed using a sub-sample X' randomly selected without replacement from X, $X' \subset X$.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - subsampling size
Output: a set of t *iTrees*

```
1: Initialize Forest
2: for i = 1 to t do
3:    X' ← sample(X, ψ)
4:    Forest ← Forest ∪ iTree(X')
5: end for
6: return Forest
```

There are two input parameters to the *i*Forest algorithm in Algorithm 1. They are the subsampling size ψ and the number of trees t; and the effects of the parameters are given below.

subsampling size ψ controls the training data size. We find that when ψ increases to a desired value, *i*Forest detects reliably and there is no need to increase ψ further because it increases processing time and memory size without any gain in detection accuracy. As we assume that anomalies are 'few' and 'different', normal points are also assumed to be 'many' and 'similar'. Under these assumptions, a small subsampling size is enough for *i*Forest to distinguish anomalies from normal points.

Empirically, we also find that setting ψ to 2^8 or 256 generally is enough to perform anomaly detection across a wide range of data. Unless otherwise specified, we use $\psi = 256$ as the default value for our experiment. An analysis on the effect of subsampling size can be found in Section 5.5 which shows that the detection

Algorithm 2 : $iTree(X')$

Inputs: X' - input data
Output: an iTree

if X' cannot be divided **then**
  return $exNode\{Size \leftarrow |X'|\}$
else
  let Q be a list of attributes in X'
  randomly select an attribute $q \in Q$
  randomly select a split point p between the max and min values of attribute q in X'
  $X_l \leftarrow filter(X', q < p)$
  $X_r \leftarrow filter(X', q \geq p)$
  return $inNode\{Left \leftarrow iTree(X_l),$
    $Right \leftarrow iTree(X_r),$
    $SplitAtt \leftarrow q,$
    $SplitValue \leftarrow p\}$
end if

performance is near optimal at this default setting and insensitive to a wide range of ψ.

Number of trees t controls the ensemble size. We find that path lengths usually converge well before $t = 100$. Unless otherwise specified, we use $t = 100$ as the default value in our experiment.

At the end of the training process, a collection of trees is returned and is ready for evaluation. The worse case time complexity of training an iForest is $O(t\psi^2)$ and the space complexity is $O(t\psi)$.

4.2 Evaluation Stage

Algorithm 3 : $PathLength(x, T, hlim, e)$

Inputs : x - an instance, T - an iTree, $hlim$ - height limit, e - current path length; to be initialized to zero when first called
Output: path length of x

if T is an external node or $e \geq hlim$ **then**
  return $e + c(T.size)$ {$c(.)$ is defined in Equation 1}
end if
$a \leftarrow T.splitAtt$
if $x_a < T.splitValue$ **then**
  return $PathLength(x, T.left, hlim, e + 1)$
else {$x_a \geq T.splitValue$}
  return $PathLength(x, T.right, hlim, e + 1)$
end if

In the evaluation stage, as implemented in Algorithm 2, a single path length $h(x)$ is derived by counting the number of edges e from the root node to an external node

as instance x traverses through an iTree. When the traversal reaches a predefined height limit $hlim$, the return value is e plus an adjustment $c(Size)$. This adjustment accounts for estimating an average path length of a random sub-tree which could be constructed using data of $Size$ beyond the tree height limit. When $h(x)$ is obtained for each tree of the ensemble, an anomaly score is computed. The anomaly score and the adjustment $c(Size)$ are to be defined in the next subsection. The worse case time complexity of the evaluation process is $O(nt\psi)$, where n is the testing data size. Details of the evaluation stage can be found in Algorithm 3.

4.3 Anomaly Score

An anomaly score is required for any anomaly detection method. The difficulty in deriving such a score from $h(x)$ is that while the maximum possible height of iTree grows in the order of ψ, the average height grows in the order of $\log \psi$. When required to visualize or compare path lengths from models of different subsampling sizes, normalization of $h(x)$ by any of the above terms either is not bounded or cannot be directly compared. Thus, a normalized anomaly score is needed for the aforementioned purposes.

iTree	BST
Proper binary trees	Proper binary trees
External node termination	Unsuccessful search
Not applicable	Successful search

Table II. List of equivalent structure and operations in iTree and Binary Search Tree (BST)

Since iTrees have an equivalent structure to Binary Search Tree or BST (see Table II), the estimation of average $h(x)$ for external node terminations is the same as that of the unsuccessful searches in BST. We borrow the analysis from BST to estimate the average path length of iTree. Given a sample set of ψ instances, Section 10.3.3 of [Preiss 1999] gives the average path length of unsuccessful searches in BST as:

$$c(\psi) = \begin{cases} 2H(\psi-1) - 2(\psi-1)/n & \text{for } \psi > 2, \\ 1 & \text{for } \psi = 2, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where $H(i)$ is the harmonic number and it can be estimated by $\ln(i) + 0.5772156649$ (Euler's constant). As $c(\psi)$ is the average of $h(x)$ given ψ, we use it to normalise $h(x)$. The anomaly score s of an instance x is defined as:

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}}, \quad (2)$$

where $E(h(x))$ is the average of $h(x)$ from a collection of iTrees. The following conditions provide three special values of the anomaly score:

(a) when $E(h(x)) \to 0$, $s \to 1$;
(b) when $E(h(x)) \to \psi - 1$, $s \to 0$; and
(c) when $E(h(x)) \to c(\psi)$, $s \to 0.5$.

Figure 4 illustrates the relationship between $E(h(x))$ and s, and the range of values are $0 < s \leq 1$ and $0 < h(x) \leq \psi - 1$.

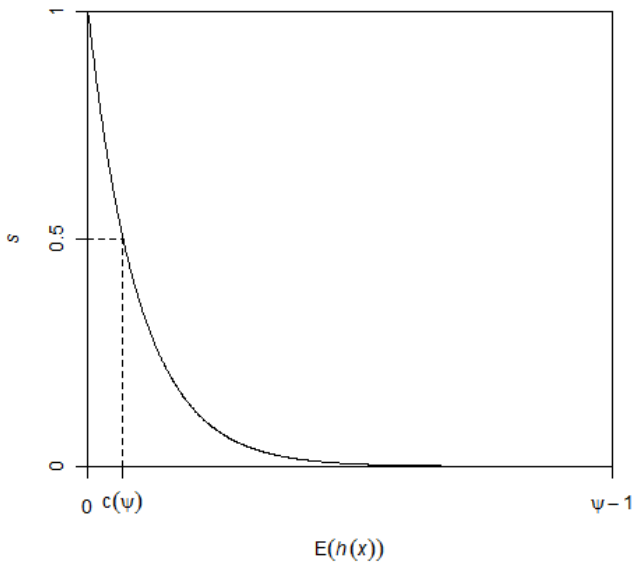


Fig. 4. The relationship of expected path length $E(h(x))$ and anomaly score s. $c(\psi)$ is the average path length as defined in equation 1. If the expected path length $E(h(x))$ is equal to the average path length $c(\psi)$, then $s = 0.5$, regardless of the value of ψ.

Using the anomaly score s, we are able to make the following assessment:

(i) if instances return s very close to 1, then they are definitely anomalies,
(ii) if instances have s much smaller than 0.5, then they are quite safe to be regarded as normal instances, and
(iii) if all the instances return $s \approx 0.5$, then the entire sample does not really have any distinct anomaly.

A contour of the anomaly score can be produced by passing a lattice sample through *i*Forest, facilitating a detailed analysis of the detection result. Figure 5 shows an example of such a contour, in order to visualise and identify anomalies in the instance space. Using the contour, we can clearly identify three points, where $s > 0.6$, which are the potential anomalies.

4.4 Small Sub-samples Build Better Isolation Models

The problems of swamping and masking have been studied extensively in anomaly detection [Murphy 1951]. Swamping refers to situations where normal instances are wrongly identifying as anomalies. It happens when the number of normal instances increases or they become more scattered. Masking, on the other hand,

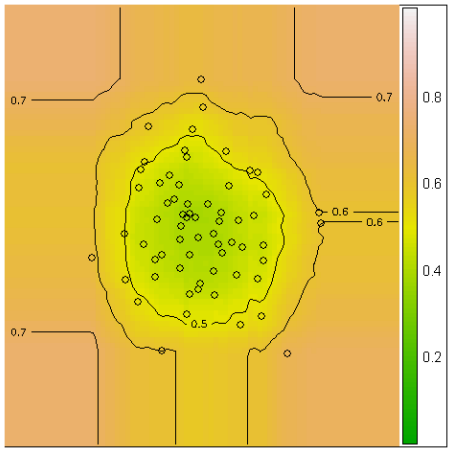
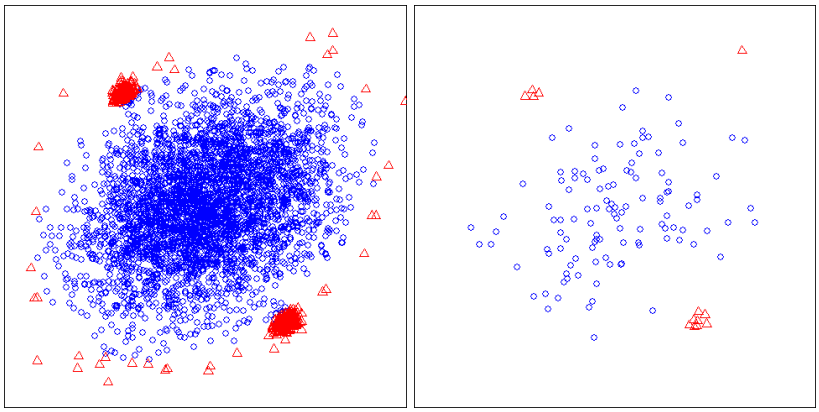


Fig. 5. Anomaly score contour of iForest for a Gaussian distribution of sixty-four points. Contour lines for $s = 0.5, 0.6, 0.7$ are illustrated. Potential anomalies can be identified as points where $s > 0.6$.

is the existence of too many anomalies concealing their own presence. It happens when anomaly clusters become large and dense. Under these circumstances, many anomaly detectors break down. Note that both swamping and masking effects are results of too many data for the purpose of anomaly detection.

iForest is able to build a model by using multiple sub-samples to reduce the effects of swamping and masking. We find that each sub-sample of a small size builds a better performing iTree than that from the entire data set. This is because sub-samples have fewer normal points 'interfering' with anomalies; thus, making anomalies easier to isolate.

To illustrate how subsampling reduces the effects of masking and swamping, Figure 6(a) shows a data set of 4096 instances generated by Mulcross data generator [Rocke and Woodruff 1996]. We deliberately set two rather large and dense anomaly clusters close to a large cluster of normal points to illustrate the effects of masking and swamping. There are interfering normal points surrounding the anomaly clusters, and the anomaly clusters are denser than the normal cluster. Figure 6(b) shows a sub-sample of 128 instances of the original data. The anomaly clusters are clearly identifiable in the sub-sample. Those 'spurious' normal instances surrounding the two anomaly clusters, which causes the swamping effect, have been cleared out; and the data size of anomaly clusters in Figure 6(a), which causes the masking effect, becomes smaller as shown in Figure 6(b). The net effect is that they make the anomaly clusters easier to isolate. When using the entire set in Figure 6(a), iForest reports an AUC of 0.67. When using a subsampling size of 128 and 256, iForest achieves an AUC of 0.91 and 0.83, respectively. The result shows that iForest is excellent in handling the effects swamping and masking through significantly reduced sub-samples. Note that anomalies are still denser than normal points even under subsampling as shown in Figure 6(b)); and they still evade detection from distance and density based methods. In the next subsection, we will see how iForest can make use of the evaluation height limit to further handle dense anomaly clusters.



(a) Original sample (4096 instances)

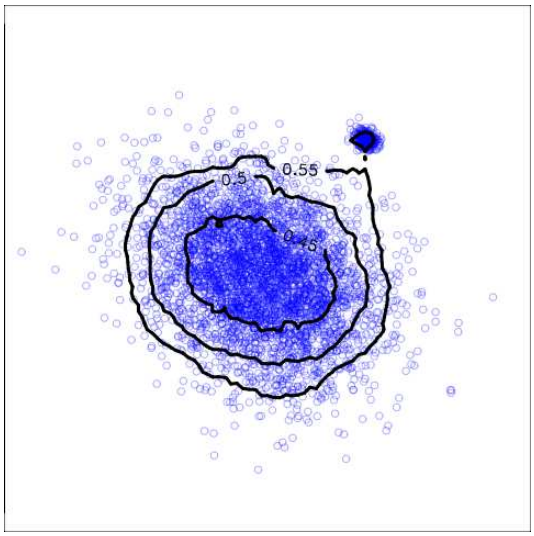
(b) Sub-sample (128 instances)

Fig. 6. Using an artificially generated data set to demonstrate the effects of swamping and masking. Figure (a) shows the original data generated by Mulcross ($n = 4096, a = 0.1, cl = 2, D = 0.8, d = 2$) and Figure (b) shows a sub-sample of the original data. Circles (○) denote normal instances and triangles (△) denote anomalies.

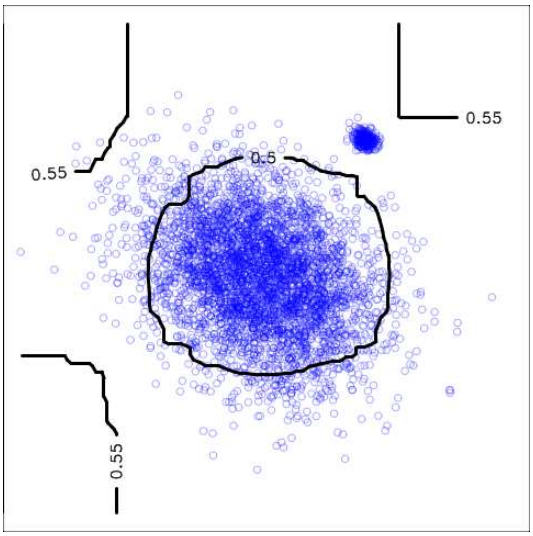
4.5 Adjusting the Granularity of Anomaly Scores

In anomaly detection, a decision needs to be made to decide whether an isolated data cluster is abnormal or its surrounding points are. We show in this section that Isolation Forest is able to detect in either case by changing the tree height limit parameter at the evaluation stage. Note that adjusting the height limit at evaluation stage does not alter the trained model and it does not require a re-training of the model. Using a data set generated by Mulcross data generator, in Figure 7, two anomaly score contours are illustrated with different height limits, i.e., $hlim = 1$ and 6. The data generated has a sparse cluster much larger than a small dense cluster. It can be noticed that at a higher height limit, i.e., $hlim = 6$, scattered points surrounding the large and small clusters have higher anomaly scores than those of the core points of both clusters. In this setting, scatter points surrounding both large and small clusters are considered as anomalies. When $hlim = 1$, the entire small dense cluster have higher anomaly scores, leading them to be identified as anomalies. The effect of using a lower height limit can be described as lowering the granularity of anomaly scores, which higher height limit provides higher granularity to detect scatter points that surround data clusters. This change of granularity brings about an advantage in detecting dense anomalies, which we shall see in Section 5.3.

In the normal usage of *i*Forest, the default value of evaluation height limit is set to maximum, i.e. $\psi - 1$, so that the anomaly score has the highest granularity. Unless otherwise specified, the evaluation tree height is set to the maximum by default throughout this paper.



(a) $hlim = 6$,
points surrounding both clusters are treated as anomalies.
Note that there is a contour line of 0.55 inside the small dense cluster.



(b) $hlim = 1$,
the small isolated cluster and the surrounding points of the large cluster are treated as anomalies.

Fig. 7. Adjusting the evaluation height limit changes the granularity of the anomaly score s which helps to modify the detection behaviour of Isolation Forest. Using a data set generated by Mulcross ($n = 4096, a = 0.06, cl = 1, D = 1, d = 2$), (a) when the evaluation height limit is set to $hlim = 6$, both the large sparse cluster (normal instances) and small dense cluster (anomalies) are encircled by contour line $s = 0.55$. Under this setting, scatter points surrounding both clusters are considered anomalies (having $s < 0.55$). (b) When $hlim = 1$, only the sparse large cluster (normal instances) has a contour line of $s = 0.5$ and there is no contour line in the small dense cluster. The small dense cluster is now considered as anomalies.

Summary

We have described the process of anomaly detection using *i*Forest—a realization of anomaly detection by isolation, and provided a meaningful anomaly score formulation based on path length. We also have analysed and explained why *i*Forest produces a better performing model using a small subsampling size and provides different levels of detection granularity by adjusting the tree height limit during the evaluation stage.

5. EMPIRICAL EVALUATION

This section presents the detailed results for seven sets of experiment designed to evaluate *i*Forest and also compare with four stat-of-the-art anomaly detectors. Section 5.1 presents the data sets used in this section. In Section 5.2, we present two experiments for benchmark evaluation. In the first experiment, two well known statistical data sets are used to verify the anomaly detection ability of each of the detectors. This experiment demonstrates the behaviours of these detectors. In the second experiment, we compare *i*Forest with other detectors using twelve data sets that are previously used to evaluate anomaly detectors in the literature. In Section 5.3, the third experiment examines the breakdown characteristic of all anomaly detectors, due to masking, under size-increasing dense anomaly clusters. We also examine the different evaluation height limits and their effects on the breakdown characteristic of *i*Forest. In Section 5.4, the fourth experiment examines the breakdown characteristic, due to swamping, when anomalies come close to normal points (swamping effect). In Section 5.5, the fifth experiment examines the impact of different subsampling sizes on *i*Forest. The results provide insights as to what subsampling size should be used and its effects on detection performance. We also demonstrate that the training time complexity is constant when the subsampling size and ensemble size are fixed. In Section 5.6, the sixth experiment investigates *i*Forest's ability to handle high-dimensional data; we reduce the attribute space before the tree construction by using a simple uni-variate test. We aim to find out whether this simple mechanism is able to improve *i*Forest's detection performance in high dimensional spaces. In many situations, anomaly data are hard to obtain; in Section 5.7, the seventh experiment examines *i*Forest's performance when only normal instances are available for training.

For all the experiments, actual CPU time and Area Under Curve (AUC) are reported. Anomaly scores from all data points are used in AUC calculation. AUC is equivalent to the probability of an anomaly detector in scoring anomalies higher than normal points. Since AUC is cutoff independent, it measures detection accuracy regardless of the number of anomalies in data sets. AUC in our experiments are calculated by a standard performance analysis package 'ROCR' [Sing et al. 2005] in R (www.r-project.org). The procedure to calculate AUC for anomaly detection is provided in Appendix E. All experiments are conducted as single-threaded jobs processed at 2.3GHz in a Linux cluster (www.vpac.org).

The four state-of-the-art anomaly detectors used in experiments are ORCA [Bay and Schwabacher 2003], one-class SVM [Schölkopf et al. 2001], LOF [Breunig et al. 2000] and Random Forests (RF) [Shi and Horvath 2006].

ORCA is a k-Nearest Neighbour (k-nn) based method, where the largest demand

of processing time comes from the distance calculation of k nearest neighbours. Using sample randomisation together with a simple pruning rule, ORCA is able to cut down the complexity of $O(n^2)$ to near linear time [Bay and Schwabacher 2003].

In ORCA, the parameter k determines the number of nearest neighbours, increasing k also increases the run time. We use ORCA's default setting of $k = 5$ in our experiments unless otherwise specified. The parameter N determines how many anomalies are to be reported. If N is small, ORCA increases the running cut-off rapidly and pruning off more searches, resulting in a much faster run time. However, it would be unreasonable to set N below the number of anomalies due to AUC's requirement to report anomaly scores for every instances. Since choosing N has an effect on run time and the number of anomalies is not supposed to be known in the training stage, we will use a reasonable value $N = \frac{n}{8}$ unless otherwise specified[2]. In reporting processing time, we report the total training and testing time, but omit the pre-processing time "dprep" from ORCA.

For one-class SVM, we use the commonly used Radial Basis Function kernel and inverse width parameter estimated by the method suggested in [Caputo et al. 2002]. For LOF, a well-known density based method, we use a commonly used setting of $k = 10$ in our experiments. We also include RF, since this is also a tree ensemble algorithm. For RF, we use $t = 100$ and other parameters in their default values. Because RF is a supervised learner, we follow the exact instruction as in [Shi and Horvath 2006] to generate synthetic data as the alternative class. The instances for the alternative class are generated by uniformly sampling random points valued between the maximum and minimum of every attribute. A proximity measure between every pair of instances is calculated after decision trees are being constructed, and anomalies are instances whose proximities to all other instances in the data are small.

For algorithms that have random elements, their results are reported using an average of ten runs. Such algorithms are ORCA, Random Forest and *i*Forest.

5.1 Data Sets

We use two statistical data sets *hbk* and *wood* [Rousseeuw and Leroy 1987] for the first experiment. As for other performance analysis and evaluation, eleven natural data sets plus a synthetic data set are used. They are selected because they contain known anomaly classes which will be used as the ground truth; and these data sets have been used in the literature to evaluate anomaly detectors in similar settings. They include: the two biggest data subsets (*Http* and *Smtp*) of KDD CUP 99 network intrusion data as used in [Yamanishi et al. 2000], *Annthyroid*, *Arrhythmia*, Wisconsin Breast Cancer (*Breastw*), Forest Cover Type (*ForestCover*), *Ionosphere*, *Pima*, *Satellite*, *Shuttle* [Asuncion and Newman 2007], *Mammography*[3] and Mulcross [Rocke and Woodruff 1996]. Their previous usages can be found in [Yamanishi et al. 2000; Abe et al. 2006; He et al. 2005; Bay and Schwabacher 2003;

[2]ORCA is able to provide an estimation of anomaly score to every point, even though N is set to be less than n. ORCA provides accurate score of the top N points, and a rough estimated score for the rest of the points. Thus, all points are used in the AUC calculation even though $N = n/8$. Using ORCA's original default setting ($k = 5$ and $N = 30$), all data sets larger than one thousand points report AUC close to 0.5, which is equivalent to randomly selecting points as anomalies.
[3]The Mammography data set was made available courtesy of Aleksandar Lazarevic.

Lazarevic and Kumar 2005; Williams et al. 2002]. Since we are only interested in continuous-valued attributes in this paper, all nominal and binary attributes are removed. The synthetic data generator, Mulcross generates a multi-variate normal distribution with a user-specified number of anomaly clusters. In our experiments, the basic setting for Mulcross is as following: number of points $n = 262144$, contamination ratio $a = 10\%$ (the number of anomalies over the total number of points), distance factor $D = 2$ (distance between the center of normal cluster and the centers of anomaly clusters), the number of dimensions $d = 4$, and number of anomaly clusters $cl = 2$. This setting is used unless otherwise stated. An example of the Mulcross data can be found in Figure 6. Table III provides the properties of all data sets and information on anomaly classes sorted by the size of data in descending order. The treatment of using minor classes as anomalies is adopted from previous works in a similar setting, e.g. [Abe et al. 2006].

	n	d	anomaly class
Http (KDDCUP99)	567497	3	attack (0.4%)
ForestCover	286048	10	class 4 (0.9%) vs. class 2
Mulcross	262144	4	2 clusters (10%)
Smtp (KDDCUP99)	95156	3	attack (0.03%)
Shuttle	49097	9	classes 2,3,5,6,7 (7%)
Mammography	11183	6	class 1 (2%)
Annthyroid	6832	6	classes 1, 2 (7%)
Satellite	6435	36	3 smallest classes (32%)
Pima	768	8	pos (35%)
Breastw	683	9	malignant (35%)
Arrhythmia	452	274	classes 03,04,05,07, 08,09,14,15 (15%)
Ionosphere	351	32	bad (36%)
hbk	75	4	14 points (19%)
wood	20	6	6 instances (30%)

Table III. Properties of the data used in the experiments, where n is the number of instances, and d is the number of dimensions, and the percentage in bracket indicates the percentage of anomalies.

It is assumed that anomaly labels are unavailable in the training stage. Anomaly labels are only used to compute the performance measure AUC after the evaluation stage.

5.2 Empirical Comparison of *i*Forest, ORCA, SVM, LOF and Random Forests

This subsection consists of two experiments. In the first experiment, using two previously used statistical data sets, *hbk* and *wood* [Rousseeuw and Leroy 1987], we examine the basic behaviour of each anomaly detector. In the second experiment, we evaluate the performance of each detector in terms of AUC and processing time.

Statistical data sets. Our first statistical data set is the *hbk* data set, it consists of 75 instances of which 14 are anomalies. Anomalies comprises of two small clusters with one more scattered than the other. Figure 8 shows the top 20 anomalies from

five anomaly detectors and a plot of the first two principle components[4] established from the original 4 attributes. The result shows that *i*Forest and RF are the only two detectors which have ranked all the anomalies at the top of the list.

rank#	iForest	Orca	SVM	LOF	RF
1	**14**	**14**	**1**	**14**	**12**
2	**12**	**12**	**3**	47	**14**
3	**13**	**11**	**4**	68	**13**
4	**11**	**13**	**7**	16	**11**
5	**10**	**7**	**10**	61	**4**
6	**4**	47	**11**	34	**9**
7	**7**	**4**	**12**	53	**1**
8	**1**	**3**	**13**	**12**	**10**
9	**2**	**1**	**14**	52	**7**
10	**8**	**10**	16	42	**6**
11	**9**	**6**	43	37	**3**
12	**3**	70	44	75	**8**
13	**5**	53	47	44	**2**
14	**6**	25	49	36	**5**
15	47	68	60	31	43
16	52	43	68	70	22
17	68	**8**	**2**	62	61
18	53	**2**	**5**	30	20
19	43	62	**6**	35	38
20	60	**5**	**8**	43	74

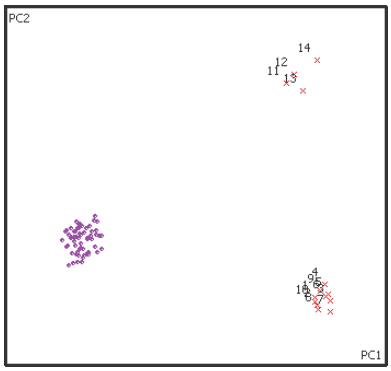


Fig. 8. (left) Anomaly ranking of the *hbk* data set, bold-faced datum indexes are actual anomalies. (right) Visualization of *hbk* data with its first two principle components, datum 1 to 10 are located at the bottom right corner as a dense cluster, datum 11 to 14 are located top right corner as scattered points.

rank#	iForest	Orca	SVM	LOF	RF
1	**19**	**19**	10	**19**	11
2	**8**	**8**	11	**8**	10
3	10	**6**	12	**6**	13
4	**4**	**4**	**19**	**4**	**8**
5	**6**	10	1	7	7
6	20	7	2	10	**4**
7	12	12	3	12	12
8	7	11	**4**	18	9
9	11	9	5	13	**19**
10	13	13	**6**	11	**6**

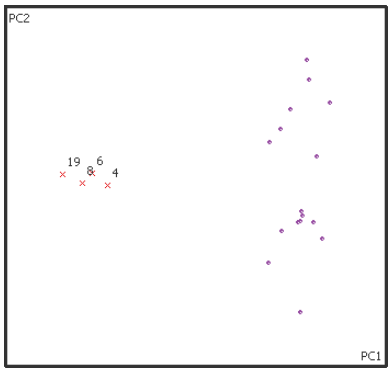


Fig. 9. (left) Anomaly ranking of the *wood* data set, bold-faced datum indexes are actual anomalies. (right) Visualization of the *wood* data with its two principle components, datum 4, 6, 8, 19 are located on the right hand side as dense points.

[4]Principle components are used for visualization purpose only; the detection results are obtained using original attributes.

	AUC				
	*i*Forest	ORCA	SVM	LOF	RF
Http (KDDCUP99)	**1.00**	0.36	0.90	*	**
ForestCover	0.87	0.83	**0.90**	0.57	**
Mulcross	**0.96**	0.33	0.59	0.59	**
Smtp (KDDCUP99)	**0.89**	0.80	0.78	0.32	**
Shuttle	**1.00**	0.60	0.79	0.55	**
Mammography	**0.84**	0.77	0.65	0.67	**
Annthyroid	**0.84**	0.68	0.63	0.72	**
Satellite	**0.73**	0.65	0.61	0.52	**
Pima	0.67	**0.71**	0.55	0.49	0.65
Breastw	**0.98**	**0.98**	0.66	0.37	0.97
Arrhythmia	**0.81**	0.78	0.71	0.73	0.60
Ionosphere	0.83	**0.92**	0.71	0.89	0.85

(a) AUC performance

	Time (seconds)						
	*i*Forest			ORCA	SVM	LOF	RF
	Train	Eval.	Total				
Http	0.25	15.33	**15.58**	9487.47	35872.09	*	**
ForestCover	0.76	15.57	**16.33**	6995.17	9737.81	224380.19	**
Mulcross	0.26	12.26	**12.52**	2512.20	7342.54	156044.13	**
Smtp	0.14	2.58	**2.72**	267.45	986.84	24280.65	**
Shuttle	0.30	2.83	**3.13**	156.66	332.09	7489.74	**
Mammography	0.16	0.50	**0.66**	4.49	10.8	14647.00	**
Annthyroid	0.15	0.36	**0.51**	2.32	4.18	72.02	**
Satellite	0.46	1.17	**1.63**	8.51	8.97	217.39	**
Pima	0.17	0.11	0.28	**0.06**	**0.06**	1.14	4.98
Breastw	0.17	0.11	0.28	**0.04**	0.07	1.77	3.10
Arrhythmia	2.12	0.86	2.98	**0.49**	0.15	6.35	2.32
Ionosphere	0.33	0.15	0.48	**0.04**	0.04	0.64	0.83

(b) Actual processing time

* Execution time takes more than two weeks

** Out of memory

Table IV. *i*Forest performs favourably to ORCA and SVM in terms of (a) AUC and (b) processing time, especially for those large data sets where $n > 1000$. Boldfaced are best performance. *i*Forest is significantly faster than ORCA and SVM for large data sets where $n > 1000$. We do not have the full results for LOF and RF because: (1) LOF has a high computation complexity and is unable to complete the largest data set within two weeks; (2) RF has a huge memory requirement, which requires system memory of $(2n)^2$ to produce proximity matrix in unsupervised learning settings.

Our second statistical data set is the *wood* data set, it consists of 20 instances of which 6 are anomalies. Anomalies are all in a smaller cluster. Figure 9 shows the top 10 instances ranked by all anomaly detectors and a two dimensional plot of two principle components of the original 6 dimensions. ORCA and LOF identify all four anomalies correctly, while *i*Forest has misranked one instance and RF performs poorly in this data set.

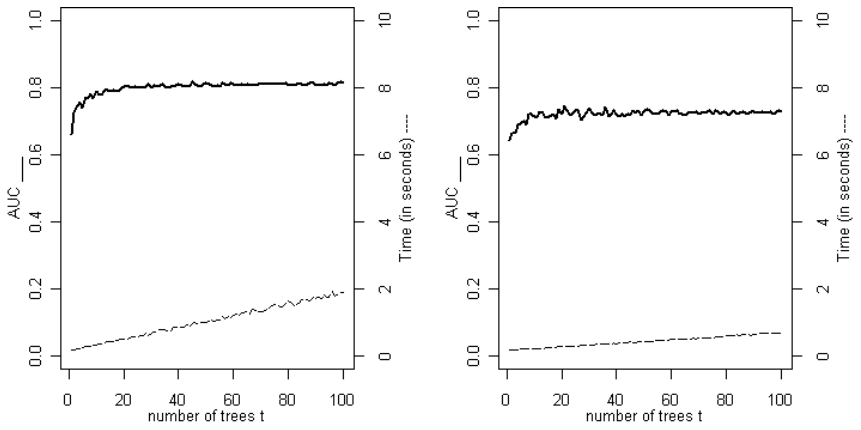
Overall, *i*Forest is the only anomaly detector that performs equally well in these two statistical data sets.

Performance Evaluation. The aim of the second experiment is to compare

*i*Forest with ORCA, LOF, SVM and RF in terms of AUC and processing time. Table IV reports the AUC score and the actual run time for all methods. From the table, we observe that *i*Forest compares favourably to all the other methods in terms of AUC and processing time. In particular, *i*Forest is fastest and the most accurate for data sets larger than one thousand points. The only exception is the *ForestCover* data set as shown in Table IV(a) in which SVM has a slightly better AUC compared to *i*Forest(0.03 difference in AUC).

Note that the difference in execution time is huge between *i*Forest, ORCA, SVM and LOF, especially in large data sets; this is due to the fact that *i*Forest is not required to compute pair-wise distances as in ORCA and LOF and kernel optimization as in SVM. Note that LOF has an extended runtime and RF has a high memory requirements; they fail to run on large data sets.

In terms of AUC, *i*Forest compares favourable to ORCA in nine out of the twelve data sets, SVM eleven out of twelve, LOF seven out of eight and RF four out of four. In terms of processing time, *i*Forest is superior in data sets larger than one thousand points as compared with ORCA and SVM and *i*Forest is better in all the data sets as compared to LOF and RF.



(a) *Arrhythmia* (b) *Satellite*

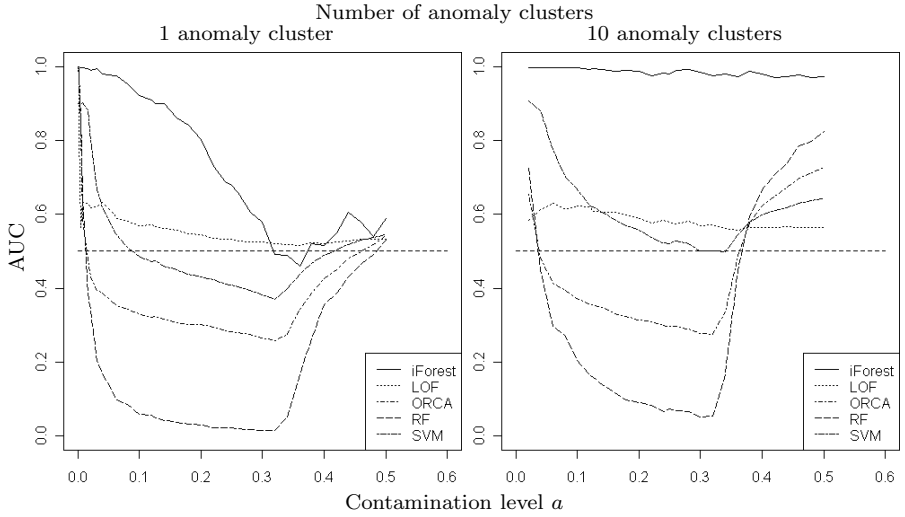
Fig. 10. Detection performance AUC (y-axis) converges at a small t (x-axis).

The performance of *i*Forest is stable in a wide range of t. Using the two data sets of the greatest dimensionality, Figure 10 shows that AUC converges at a small t. The full result is available at Appendix B. Since increasing t also increases processing time, the early convergence of AUC suggests that *i*Forest's execution time reported earlier can be further reduced if t is tuned to a data set.

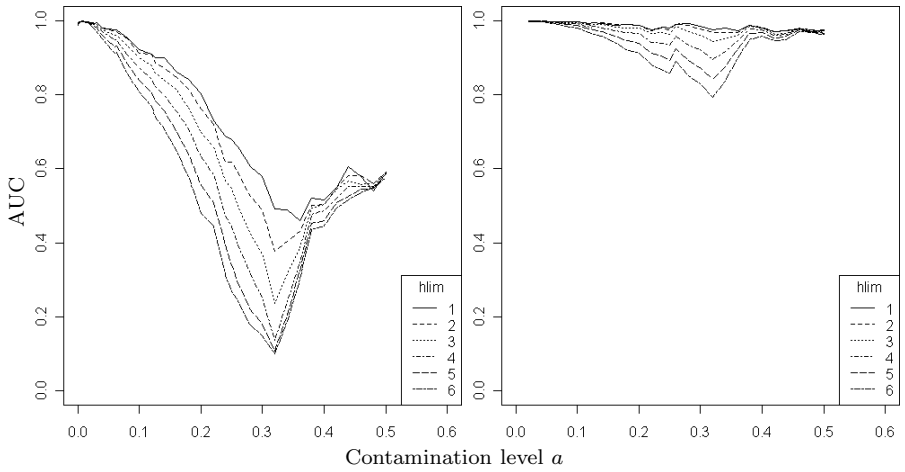
As for the *Http* and *Mulcross* data sets, due to the large anomaly-cluster size and the fact that anomaly clusters have an equal or higher density as compared to the normal instances (i.e., the masking effect), ORCA reports a result poorer than random guessing on these data sets. We also experiment ORCA on these data sets using a much higher value of k (where $k = 150$), however the detection performance is similar. This highlights a problematic assumption in ORCA and the other similar

k-nn based methods: they can only detect low-density anomaly clusters of data size smaller than k. Increasing k may solve the problem, but it is not practical in high volume setting due to the increase in the processing time and the need to find an appropriate k.

5.3 Breakdown Analysis with data-size-increasing Anomaly Clusters (Masking Effect)



(a) iForest (hlim=1) as compared to the four anomaly detectors.



(b) iForest performance on various tree height limits.

Fig. 11. AUC performance (y-axis) of five anomaly detectors on Mulcross ($n = 4096 * (1 + a), a = \{0.02, ..., 0.48, 0.5\}, cl = \{1, 4\}, D = 1, d = 2$), data under various contamination level (x-axis) with 1 and 10 anomaly clusters.

In this subsection, we examine iForest's breakdown characteristic in conjunction with its ability to alter the detection behaviour by setting different evaluation height

limits. For comparison, we also examine the breakdown characteristics of the four anomaly detectors introduced in the beginning of this section.

Using the Mulcross data generator to generate 4096 normal instances with anomalies of different contamination levels and different numbers of anomaly clusters, we examine the detection accuracy (AUC) of all the detectors. AUC generally drops as the contamination level increases—the rate the AUC drops indicates how quickly an anomaly detector breaks down under the condition. The contamination level ranges between $a = 0.02$ and $a = 0.5$.

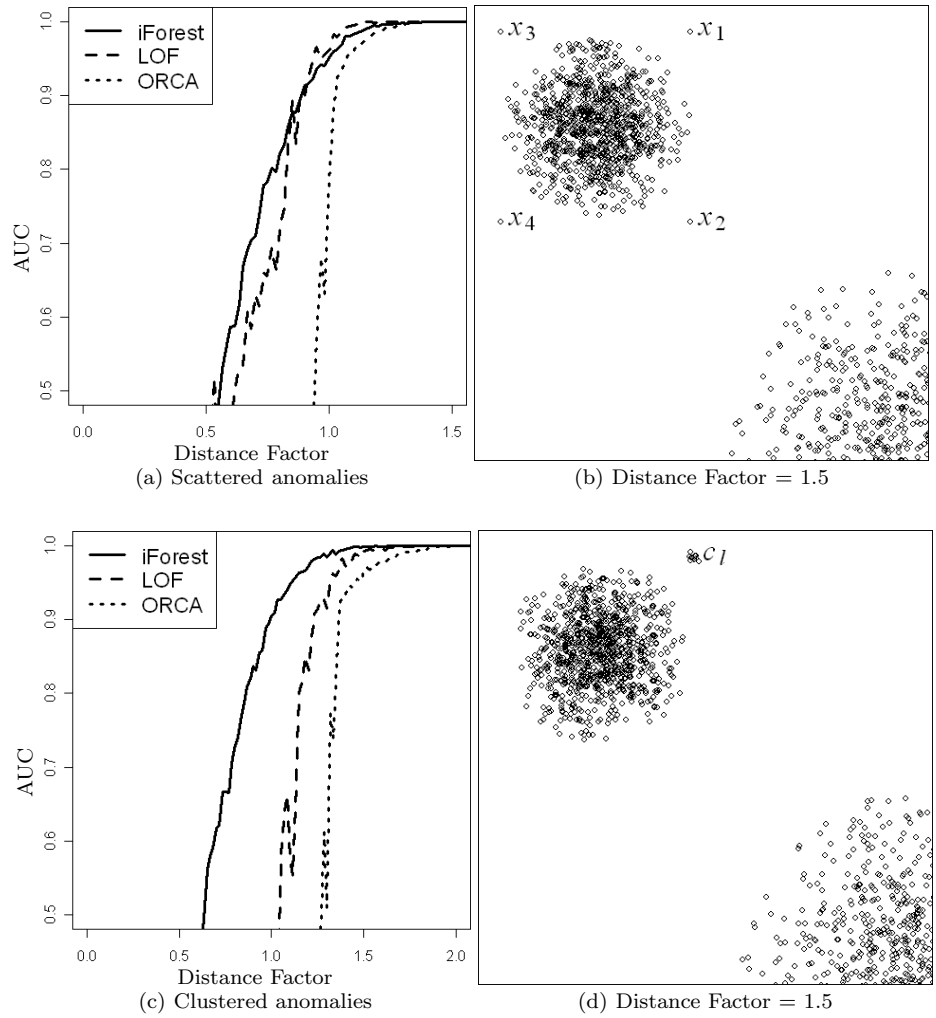
In Figure 11(a), all the other four anomaly detectors have significantly lower detection performance than *i*Forest throughout the entire range of contamination, though some hold up better than the others as the contamination level increases. The dense anomaly clusters mislead the other anomaly detectors more than *i*Forest—most of them break down with very low detection performance before the contamination level even reaches $a = 0.1$; *i*Forest is more robust and it breaks downs more gradually as the contamination level increases. Note that the problem with ten anomaly clusters is an easier problem than the one with only one anomaly cluster. It is because the same number of anomalies are now grouped into ten clusters rather than concentrated on the one cluster at the same contamination level. *i*Forest almost maintains its detection performance for the entire contamination range. However, this only has a marginal effect on the other four detectors, the high volume and high density of the anomaly clusters still pose difficulties.

For *i*Forest, in Figure 11(b), we find that setting a lower evaluation height limit is effective in handling dense anomaly clusters. *i*Forest obtains its best performance using $hlim = 1$. It is because *i*Forest uses the coarsest granularity to detect clustered anomalies.

5.4 Breakdown Analysis with Local Anomalies (Swamping Effect)

When anomalies become too close to normal instances, anomaly detectors break down due to the proximity of anomalies. To examine the robustness of different detectors against local anomalies, we generate anomalies with various distances from a normal cluster in the context of two normal clusters of different densities. We use a distance factor $= \frac{l}{r}$, where l is the distance between anomaly cluster and the center of a normal cluster and r is the radius of the normal cluster. When the distance factor equals to one, the anomaly cluster is located right at the edge of the dense normal cluster. In this evaluation, LOF and ORCA are given $k = 15$ so that k is larger than the size of largest anomaly cluster. Random Forest and SVM do not do well in this analysis. For clarity, only the top three detectors are shown in this analysis.

As shown in Figure 12(a), when anomalies are scattered, *i*Forest has a similar performance as compared with LOF, followed by ORCA. When anomalies are clustered, as shown in Figure 12(c) and (d), *i*Forest clearly has a better performance than LOF and ORCA. Figure 12(b) and (d) show both scenarios with the distance factor = 1.5. The outstanding performance of *i*Forest in clustered anomalies is attributed to the use of isolation, which covers the context of the entire data set as well as the context of local neighbourhood. When clustered anomalies are close to the normal cluster, it is hard to detect them, i.e., they are not too far from their neighbours and their relative densities are similar to other points. However, these



(a) Scattered anomalies (b) Distance Factor = 1.5

(c) Clustered anomalies (d) Distance Factor = 1.5

Fig. 12. Performance in detecting Local Anomalies. Results are shown in (a) and (c) with AUC (y-axis) versus distance factor (x-axis). (b) and (d) illustrate the data distributions in both scattered and clustered anomales when distance factor = 1.5.

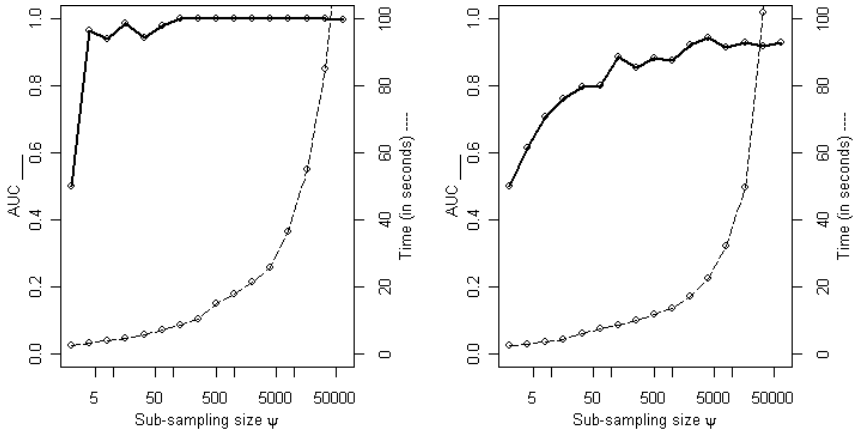
clustered anomalies are still distinguishable in the context of the entire data set. That is why *i*Forest is able to detect clustered anomalies well.

5.5 Analysis on the effect of subsampling

The following experiment investigates *i*Forest's efficiency in terms of the memory requirement and training time, in relation to the subsampling size ψ. In addition, we also examine how well the detectors perform with limited supply of data. In this experiment we adjust the subsampling size in the range of $\psi = 2, 4, 8, 16, ..., 65536$.

Two examples of our findings are shown in Figure 13, we observe that the AUC of *i*Forest converges very quickly at a small ψ. AUC is near optimal when $\psi = 128$ for Http and $\psi = 512$ for ForestCover, and they are only a fraction of the original

data (0.00045 for Http and 0.0018 for ForestCover), which makes *i*Forest highly efficient using a small ψ. For larger ψs, the variation of AUC is minimal: ± 0.0014 and ± 0.023 respectively. Also note that the processing time increases very modestly when ψ increases from 4 up to 8192. *i*Forest maintains its near optimal detection performance within this range. It shows that the detection performance is mainly due to the proposed tree-based algorithm and not subsampling. The increase of ψ has an adverse effect on Mulcross generated data, after $\psi > 64$. It is due to the masking effect of the dense anomaly clusters in Mulcross. As explained in Section 4.4, masking effect can be handled by reduced sub-samples in *i*Forest. In a nutshell, high detection performance is a characteristic of *i*Forest and using a small ψ results in low processing time, and a further increase of ψ is not necessary. For a complete result of all data sets, please see Appendix C.



(a) *Http* (b) *ForestCover*

Fig. 13. A small subsampling size provides both high AUC (left y-axis, solid lines) and low processing time (right y-axis, dashed lines, in seconds). subsampling size (x-axis, log scale) ranges $\psi = 2, 4, 8, 16, ..., 65536$.

The implication of using a small sub-sample size is that one can easily host an online anomaly detection system with a minimal memory footprint. Using $\psi = 256$, the maximum number of nodes is 511. Let the memory requirement of a node be b bytes, t be the number of trees. Thus, a working model to detect anomalies is estimated to be less than $511tb$ bytes, which is trivial in modern computing equipments. It is important to notice that the memory requirement is constant using fixed number of trees and subsampling size.

As for training time, when we use a constant $\psi = 256$ on the Mulcross generated data of different sizes, *i*Forest's training time grows less than half when the size of the data doubles as reported in Table V. Discounting the operations of loading the training data into the memory and the subsampling process, the time that took to construct $t = 100$ trees is constant at approximately 0.04 second. The total processing time of *i*Forest grows in the same rate as the data size only when the data size becomes very large; otherwise it is sub-linear. In contrast, both ORCA and SVM have their execution times grow about four times when data size double.

	AUC			Time (seconds)				
				*i*Forest				
n	*i*Forest	ORCA	SVM	Train	Eval	Total	ORCA	SVM
1024	0.90	0.23	0.59	0.152	0.061	0.213	0.08	0.10
2048	0.89	0.28	0.60	0.151	0.077	0.228	0.21	0.68
4096	0.91	0.31	0.59	0.156	0.122	0.278	0.73	2.61
8192	0.90	0.33	0.49	0.162	0.201	0.363	2.63	9.59
16384	0.89	0.33	0.58	0.170	0.364	0.534	10.19	37.04
32768	0.91	0.33	0.59	0.182	0.775	0.957	40.20	152.82
65536	0.91	0.33	0.59	0.199	1.510	1.709	158.16	551.88
131072	0.95	0.33	0.59	0.227	2.948	3.175	637.83	2040.63
262144	0.94	0.33	0.59	0.272	5.970	6.242	2535.60	7899.46
524288	0.95	0.33	0.49	0.359	14.144	14.503	10203.64	28737.42
1048576	0.94	0.33	0.59	0.577	23.863	24.440	40779.85	155084.49

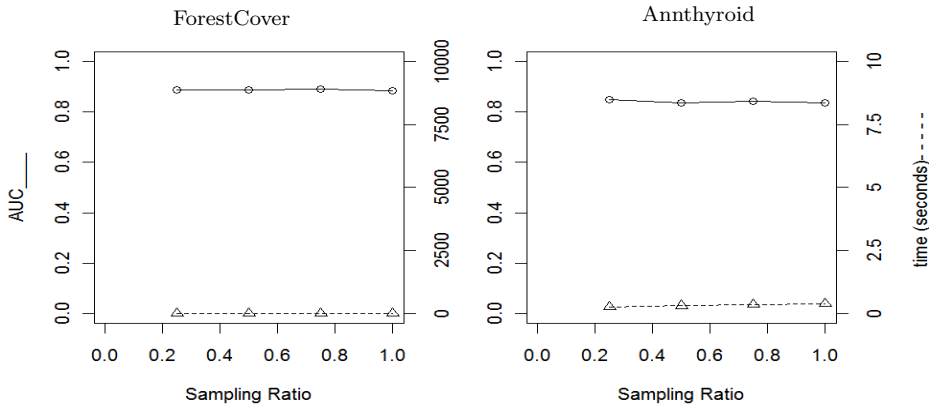
Table V. Performance of *i*Forest, ORCA and SVM on Mulcross ($n = \{1024, ..., 1048576\}, d = 4, cl = 2, D = 2, a = 0.1$) with various data sizes.

It is a significantly difference when the data size is large, e.g., with one million data points, *i*Forest takes less than one-fifteen hundredth of that from ORCA and one-six thousandth of that from SVM; and the detection accuracy is a lot better too. Note that using a smaller data size does not improve the detection result of ORCA and SVM. When using constant ψ and t, the runtime and memory requirement to construct *i*Forest is basically constant regardless of the training data size.

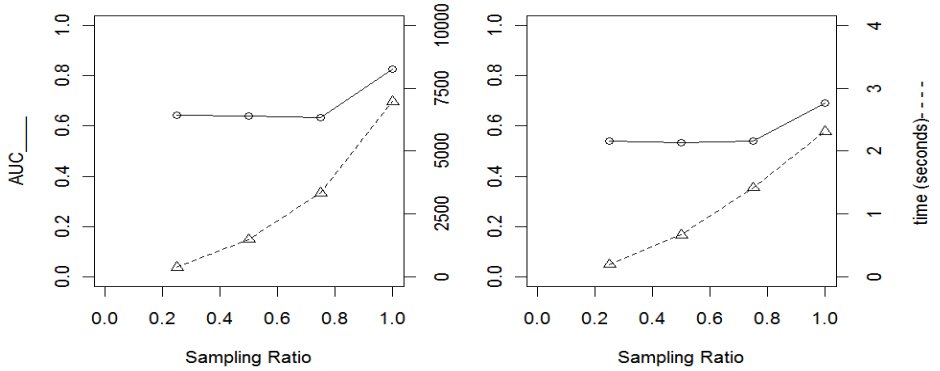
In addition, we examine the effect of sub-sample on the different anomalies detectors, with respect to AUC and processing time. The neighbourhood of every instance changes dramatically from one subsample to another; this is likely to affect the performance of k nearest neighbours (k-nn) based algorithms. With a reduced data size in a subsample, we can expect the processing time to decrease. We demonstrate the effect of subsampling using two data sets in the following paragraphs.

In Figure 14, the performances of three detectors including (a) *i*Forest, (b) ORCA and (c) SVM, are reported over different subsample sizes of the original data set in terms of four sampling ratios (0.25, 0.5, 0.75 and 1.0). We observe that the detection performances of *i*Forest and SVM are stable across various ratios. However, ORCA's detection performance deteriorates when the sampling ratio reduces from 1 to 0.75. The detection performance as shown in ORCA is analogous to the learning curve commonly found in classification models, where the reduction of training data generally results in poorer classification performance. The same is reported by [Wu and Jermaine 2006]. Because SVM constructs a boundary for normal points, it is able to construct one with a small sample; thus SVM has reached the best detection performance with sampling ratio 0.25. However, its best performance is not competitive, as in the case of Annthyroid.

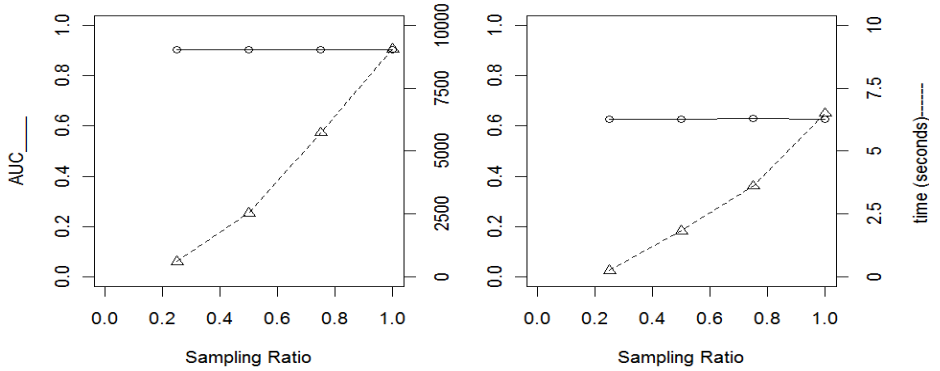
In terms of processing time, note that ORCA and SVM have a super linear increase in their processing times as the subsampling size increases, while *i*Forest has a linear increase in ForestCover and a sub-linear increase in Annthyroid. The above result suggests that *i*Forest is suitable for large data sets because it has a linear or sub-linear increase in processing time and stable detection performance across different data sizes. Note that the time complexity of *i*Forest (for training



(a) *i*Forest



(b) ORCA



(c) SVM

Fig. 14. This figure shows the detection and runtime performances of three detectors under four sampling ratios: 0.25, 0.5, 0.75 and 1 of the original data set. Samples are stratified to make sure the proportion of anomalies is maintained in the small sample. Ten-run averages are reported. Note that ORCA's detection performance deteriorates when using subsamples of the original data.

and evaluating) is $O(t\psi(\psi + n))$. For large data sets, where $n \gg \psi$, the time complexity is dominated by n.

5.6 High Dimensional Data

One of the important challenges in anomaly detection is handling high dimensional data. For distance-based or density-based methods, every point is equally sparse in high dimensional space—rendering distance a useless measure (note that many density based methods use distance to estimate density). Without any assistance, *i*Forest also suffers from the same 'curse of dimensionality'. However, using a simple attribute selector, we will see whether we can improve *i*Forest's ability to handle high dimensional data.

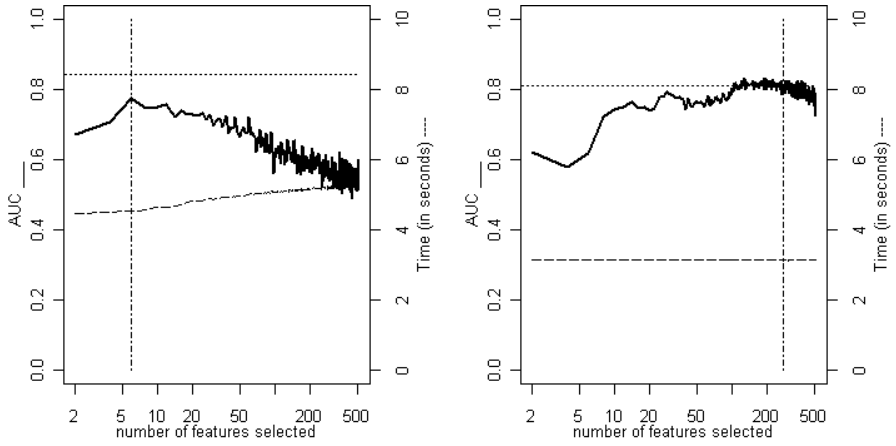
In this experiment, we study a special case of high dimensional data in which data sets have a large number of irrelevant attributes. We show that *i*Forest has a significant advantage in processing time. We simulate these high dimensional data sets using the first thirteen data sets introduced in Table III. For each data set, uniformly distributed random attributes, valued between 0 and 1 are added. Such that, there is a total of 512 attributes in each data set. We use a simple statistical test, Kurtosis [Joanes and Gill 1998], to select an attribute subspace from the sub-sample before constructing each *i*Tree. Kurtosis measures the 'peakness' of a univariate distribution. Kurtosis is sensitive to the presence of anomalies and hence it is a good attribute selector for anomaly detection. After Kurtosis has provided a ranking for each attribute, a subspace of attributes is selected according to this ranking to construct each tree. The result is promising and we show that the detection performance improves when the subspace size comes close to the original number of attributes. There are other attribute selectors that we can choose from, e.g., Grubb's test. However, in this section, we are only concern with showcasing *i*Forest's ability to work with an attribute selector in reducing the dimensionality of the anomaly detection tasks.

As an illustration, Figure 15 shows that a) the processing time of *i*Forest in Mammography and Arrhythmia remains less than 10 seconds for the whole range of subspace sizes and b) AUC peaks when subspace size equals the number of original attributes. The full result is available in Appendix D.

When ORCA is used on these two high dimensional data sets, it reports an AUC close to 0.5 and a processing time of over one hundred seconds. It shows that these high dimensional data sets are challenging, however, *i*Forest is able to improve the detection performance by a simple addition of Kurtosis test on small sub-samples rather than the entire data set. It may well be possible for other methods to apply similar attribute reduction technique to improve detection accuracy on high-dimensional data, but they would need to perform the test on the whole data set. An added advantage of *i*Forest is its low processing time even in high dimensional data.

5.7 Training Using Normal Instances Only

"Does *i*Forest work when training set contains normal instances only?" To answer this question, we remove anomalies from the training sample and evaluate the trained model with both anomalies and normal instances. We report the average AUC in Table VI. The result shows that the inclusion of anomalies make no or



(a) *Mammography (6 original attributes)* (b) *Arrhythmia (274 original attributes)*

Fig. 15. *i*Forest achieves good results on high dimensional data using Kurtosis to select attributes. Irrelevant attributes are added so the the total number of attributes reaches 512. AUC (left y-axis, solid lines) improves when the subspace size (x-axis) comes close to the number of original attributes, and the processing time (right y-axis, dashed lines, in seconds) increases slightly as subspace size increases. Training *i*Forest using the original data has a slightly better AUC (shown as the top dotted lines). Dotted dash lines indicate the original number of attributes.

little difference to the detection performance.

The reason why *i*Forest is still able to detect anomalies even in their absence in the training set is that *i*Forest describes data distribution as seen in Appendix A in which high path length values correspond to the presence of data points. Thus, the presence of anomalies is irrelevant to *i*Forest's detection performance—one less thing to worry about when using *i*Forest.

	Trained with Normal instances only	Trained with Normal instances and anomalies
Http	0.99	**1.00**
ForestCover	0.82	**0.87**
Mulcross	**1.00**	0.96
Smtp	0.88	**0.89**
Shuttle	**1.00**	**1.00**
Mammography	**0.89**	0.84
Annthyroid	**0.90**	0.84
Satellite	**0.78**	0.73
Pima	**0.73**	0.67
Breastw	**1.00**	0.98
Arrhythmia	**0.82**	0.81
Ionosphere	($\psi = 128$) **0.91**	0.83

Table VI. Comparing the AUC performance of *i*Forest trained with normal instances only versus *i*Forest trained with both the normal instances and anomalies. Better performance are boldfaced.

6. RELATED WORK

The development of anomaly detection can be divided into three major approaches; they are the density-based, distance-based and model-based approaches. Under each approach, methods share a common principle in how anomalies are detected. Note that it is possible that some methods be categorised into more than one approach. The purpose of the rest of this section is to contrast the isolation-based approach among these three approaches. Our coverage is not meant to be exhaustive, however, we provide our view of the development which helps to position the isolation-based approach. For a good survey on recent developments in anomaly detection, please see [Chandola et al. 2009] for more details.

6.1 Density-based approach

Density-based approach operates on the principle that instances in low density regions are considered as anomalies. There are many ways to estimate density, so there are many density-based methods. Well-known methods in density based approach are *Local Outlier Factor* (LOF) [Breunig et al. 2000], *Connectivity-based Outlier Factor* (COF) [Tang et al. 2002], *LOcal Correlation Integral* (LOCI) [Papadimitriou et al. 2003] and Resolution-based Outlier Factor (ROF) [Fan et al. 2006].

In LOF [Breunig et al. 2000], a LOF value is computed for each instance. The LOF value indicates the sparseness of a point in relation to its local neighbourhood. Instances with the highest LOF values are considered as anomalies.

COF [Tang et al. 2002] enhances LOF by taking into account the connectivity of the neighbours of a point, in addition to the densities of its neighbours. The anomaly score is calculated using the ratio of the average distance from the point to its k-distance neighbours and the average distance from its k-distance neighbours to their own k-distance neighbours. Tang et al. [2002] separate the treatment of low-density points and isolated points. COF defines isolativity as the degree of disconnectivity to other neighbouring points. High isolativity implies low density, however, low density does not always imply high isolativity. Isolativity is different from the 'isolation' concept we proposed in this paper, because isolativity is defined by density and isolation is not.

Papadimitriou et al. [2003] propose LOCI based on multi-granularity deviation factor (MDEF). MDEF measures the relative deviation of density of a point's neighbourhood in order to detect anomaly clusters.

ROF [Fan et al. 2006] defines anomaly as a point which is inconsistent with the majority of the data at different resolutions. ROF is defined as the accumulated ratio of the sizes of clusters containing a point in two consecutive resolutions. Anomalies are data points with the lowest ROF values.

K-d Tree [Chaudhary et al. 2002] is a tree-based density model, which partitions instance space into rectangular regions. Each region has nearly uniform sparseness and the degree of anomaly of a region is measured by the relative sparseness to its nearby regions. Since sparseness is the inverse of density, this method still requires density calculation for each region. Different from iForest, K-d Tree is a single tree implementation that utilizes density instead of path length to detect anomalies.

Feature Bagging [Lazarevic and Kumar 2005] is proposed to improve LOF, in

which data from single attributes are used to generate multiple models and the final anomaly scores are aggregated from multiple models. The improvement is marginal with the largest AUC gain of 0.025 among the six data sets reported.

The development of density-based approach is marked by an increasing number of derivatives of the original notion of density. Density-based methods introduce more and more sophisticated definitions to handle data sets of diverse density, and to widen the scope from scattered anomalies to anomaly clusters. Since density estimation is usually costly, efficiency is never a strength for density-based methods. The density-based approach seems to stagnate due to efficiency issues. Also, density definitions are based on the full dimensional distance calculation between two points, which is subjected to the curse of dimensionality.

6.2 Distance-based approach

The working principle of the distance-based approach is to identify anomalies as points that are distant from its neighbours. Euclidean distance is commonly used as the distance measure in distance-based methods. Examples are $DB(p, D)$ [Knorr and Ng 1998; Knorr et al. 2000], D_n^k [Ramaswamy et al. 2000], ORCA [Bay and Schwabacher 2003] and DOLPHIN [Angiulli and Fassetti 2009].

Knorr et al. [2000] and Knorr and Ng [1998] define "*a point x in a data set X is a $DB(g, D)$ outlier if at least a fraction g of the points in X lies at a greater distance than D from x*". Their focus was to speed up the distance calculation and proposed three algorithms based on the same anomaly definition, i.e., index-based, nested-loop and cell-based algorithms. The former two algorithms have the time-complexity of $O(n^2d)$, the last one has the time-complexity of $O(c^d + n)$, where c is a constant. $DB(p, D)$ is sensitive to the parameters D and p, and it does not provide a ranking on anomalies.

Ramaswamy et al. [2000] modify the definition in [Knorr et al. 2000] based on the distance of k^{th} nearest neighbour. This provides a ranking for a predefined number of anomalies and simplifies the user-defined parameter to a single k. Ramaswamy et al. [2000] also optimize the index-based and nested-loop algorithms and introduce a partition-based algorithm to prune off unnecessary distance calculations.

ORCA [Bay and Schwabacher 2003] is an optimized nested-loop algorithm that has near linear time complexity. ORCA randomizes the data and partitions them into blocks. It keeps track of a set of user-defined number of data points as potential anomalies along with their anomaly scores. The minimum anomaly score of the set is used as a cut-off. The cut-off is updated if there is a point with a higher score in other blocks. If a point has a lower score than the cut-off, the point will be pruned. This pruning process only speeds up the distance calculation if the ordering of data is uncorrelated. ORCA's worse case time-complexity is still $O(n^2)$ and the I/O cost is quadratic [Tao et al. 2006]. ORCA can use an anomaly definition of either k^{th} nearest neighbour or average distance of k nearest neighbours.

DOLPHIN [Angiulli and Fassetti 2009] has a time-complexity of $O(\frac{k}{p}nd)$ linear to the number of data points n, where p is the probability of randomly picking a point from a dataset that is a neighbour of a point in the index. With exactly two data scans, DOLPHIN only utilizes $\frac{k}{p}$ amount of memory. However, DOLPHIN degenerates to $O(n^2d)$ when $\frac{k}{p}$ is as large as n. Empirically, DOLPHIN compares

favourable to ORCA in terms of the execution time and I/O cost.

The development of distance-based approach is marked by a number of improvements to speed up the calculation of distance with little or no change to its anomaly definition. As a result, distance-based methods would still have trouble handling dense anomaly clusters and data sets with diverse densities. Inherently distance-based methods rely on pair-wise distance measured in full dimensional space, which is computationally expensive in high dimensional data sets.

6.3 Model-based approach

To detect anomalies, most existing model-based methods construct a model of the data, then identify the anomalies as the data points that do not fit the model well [Tan et al. 2005]. Notable examples are: classification-based methods [Abe et al. 2006], Replicator Neural Network (RNN) [Williams et al. 2002], one-class SVM [Tax and Duin 2004], clustering-based methods [He et al. 2003], link-based methods [Ghoting et al. 2004] and Convex peeling [Rousseeuw and Leroy 1987]; they all use this general principle.

In classification-based methods [Abe et al. 2006], when anomaly class is not known, the general approach is to first convert an unsupervised anomaly detection problem into a supervised classification problem by injecting artificial instances as a "background" class, which is the complement of the normal class. The background class is randomly generated samples of data instances so that the decision boundary between the normal class and the background class can be learned. Utilizing the generalization ability, the classifiers are able to detect anomalies as instances in the background class. Random Forests [Shi and Horvath 2006] as used in this paper is a classification-based method; it utilizes a proximity matrix in addition to a classifier to detect anomalies. Random Forests has a $O(n^2)$ space complexity because the size of proximity matrix is $2n \times 2n$.

In RNN [Williams et al. 2002], the anomaly detection ability comes from RNN's "inability" to reconstruct some of the data points using a neural network. Those that are poorly reconstructed are deemed anomalies.

In one-class SVM [Tax and Duin 2004], the aim is to find the smallest region that contains most of the normal data points; the points outside of this region are deemed anomalies.

In clustering-based methods [He et al. 2003], data are first clustered by a clustering algorithm; then, any small clusters or points that are distant from large cluster centroids are deemed anomalies.

LOADED [Ghoting et al. 2004] employs the idea from link analysis that data similar to each other have more "supports", and anomalies are those that have less support.

Convex peeling [Rousseeuw and Leroy 1987] is a depth-based method using ideas from the computational geometry. Using the definition of the half-space depth [Tukey 1977], each data point is assigned a depth, anomalies are points that have smaller depth values. Convex peeling is only suitable for up to 2-dimensional data. In general, depth-based methods measure how deep a point is with reference to a single data cloud [Liu et al. 1999]. In contrast, the isolation-based approach measures how isolated a point is without any assumption on the data distribution and it is able to handle data with multiple data clouds.

Most of the model-based methods suffer from the fact that their anomaly detection abilities are by-products of other algorithms designed for other purposes. The development of model-based methods seem to be ad hoc and without continuity. Their detection performance hinged very much on how well the data fit into their assumptions.

The isolation-based approach can be considered as a model-based approach, however, isolation-based approach is specially designed for anomaly detection. Unlike density and distance approaches, the definition of anomaly based on isolation covers both scattered anomalies as well as clustered anomalies. The first isolation method, *i*Forest, has a very simple design and yet we have showed that the efficiency of this method is very competitive even before any attempt to speed up the processing time. *i*Forest has a time complexity of $O(t(n + \psi)\psi)$ and a space complexity of $O(t\psi)$; for large data sets, $t\psi^2 \ll n$ is a very small constant making *i*Forest an algorithm of low linear time-complexity with a low memory requirement.

6.4 Miscellaneous

*i*Forest is fundamentally different from popular tree-ensemble-based classifiers, for examples, Variable Random Trees [Liu et al. 2008b] and Random Forests [Breiman 2001]. Although they share very similar data structure. The main purpose of using tree structures in classification is to separate different classes and utilize the class labels at leaf nodes for classification. *i*Forest however only uses the tree structures as a means to isolate the instances without the class information and only the path length information is utilized.

Yu et al [2009] have investigated a deterministic isolation tree on the premise that *i*Forest has inferior detection performance with comparison to LOF. However, we have showed in a technical report [Liu et al. 2010a] that *i*Forest and LOF have similar detection performance in terms of AUC in the artificial data sets they have employed. Using real-world data sets, our result presented in Table IV(a) shows that *i*Forest outperforms LOF in ten out of eleven data sets, which contradicts this premise.

We have showed that *i*Forest can detect global clustered anomalies in this paper. However, *i*Forest fails to detect local clustered anomalies. SCiForest [Liu et al. 2010b], a deterministic variant of *i*Forest, is able to detect local clustered anomalies. SC*i*Forest is different from *i*Forest in that models are constructed using hyper-planes and split points are selected deterministically. Using hyperplanes and the deterministic selection criterion increases training time. Thus, it is only recommended when local clustered anomalies are present in the data.

Previous analyses [Knuth 2006; Ruskey 1980] in generating all possible trees provide a probabilistic explanation why "fringes points have markedly shorter path lengths than those from the core points." These analyses are summarized in Appendix A. However, none of the previous analyses conceive isolation as a means to detect anomalies. In addition, we show that isolation trees have this property without generating all trees—in fact, only a small number of trees generated from small sub-samples is required.

7. FUTURE WORK

*i*Forest may be extended to deal with categorical data, online anomaly detection and high dimensional data.

Unlike continuous-valued data, categorical data have no ordering information and have limited possible values. Choosing a split becomes a problem under this situation. Existing treatment that assigns possible values into different bins [Quinlan 1993] is unlikely to work well with isolation-based methods. Furthermore, in situations with mixed data types in an isolation model, categorical attributes will have a lesser impact as compared to continuous-valued attributes simply because the number of possible values in a categorical attribute can be significantly less. This would have an undesirable effect on detection performance when categorical attributes are relevant. To properly handle categorical data in isolation models is an area of further investigation.

Online applications in data streams, with potentially infinite data, demand an one-pass algorithm which is able to adapt to concept drift in evolving data streams. *i*Forest is likely to be applicable to online anomaly detection. It can start detecting early in a data stream and easily adapt to concept drift because of using small sub-samples in building a model. We expect that this line of research will be very fruitful.

8. CONCLUSIONS

This paper proposes the first isolation method, *i*Forest, and makes three key contributions in the area of anomaly detection. Firstly, we introduce the use of isolation as a more effective and efficient means to detect anomalies than the commonly used basic distance and density measures. We show that the concept of isolation, when implemented in a tree structure, takes full advantage of anomalies' properties of 'few and different' that isolates anomalies closer to the root node as compared to the normal points. This enables a profile of the instance space to be constructed using path length.

Secondly, *i*Forest is an algorithm with a low linear time complexity and a small memory requirement. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of how large a data set is. It has constant time and space complexities during training.

Thirdly, our empirical comparison with four state-of-the-art anomaly detectors shows that *i*Forest is superior in terms of:

—runtime, detection accuracy and memory requirement, especially in large data sets,

—robustness with i) the masking and swamping effects, and ii) clustered anomalies, and

—its ability to deal with high dimensional data with irrelevant attributes.

We also show that *i*Forest is capable of:

—being trained with or without anomalies in the training data, and

—providing detection results with different levels of granularity without re-training.

Acknowledgements

The authors thank Victorian Partnership for Advanced Computing (www.vpac.org) for providing the high-performing computing facility.

A preliminary version of this article appears in the Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM '08) [Liu et al. 2008a]. It also won the Runner-up Best Theoretical/Algorithms Paper Award in that conference.

Software download

The implementation of *i*Forest can be found in the following URL: `https://sourceforge.net/projects/iforest/`

REFERENCES

Abe, N., Zadrozny, B., and Langford, J. 2006. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, Philadelphia, PA, USA, 504–509.

Angiulli, F. and Fassetti, F. 2009. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. *ACM Trans. Knowl. Discov. Data 3,* 1, 1–57.

Asuncion, A. and Newman, D. 2007. UCI machine learning repository.

Bay, S. D. and Schwabacher, M. 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, Washington, D.C., 29–38.

Breiman, L. 2001. Random forests. *Machine Learning 45,* 1, 5–32.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. 2000. LOF: identifying density-based local outliers. *ACM SIGMOD Record 29,* 2, 93–104.

Caputo, B., Sim, K., Furesjo, F., and Smola, A. 2002. Appearance-based object recognition using SVMs: which kernel should I use? In *Proc of NIPS workshop on Statitsical methods for computational experiments in visual processing and computer vision*.

Chandola, V., Banerjee, A., and Kumar, V. 2009. Anomaly detection: A survey. *ACM Computing Surveys 41,* 3, 1–58.

Chaudhary, A., Szalay, A. S., Szalay, E. S., and Moore, A. W. 2002. Very fast outlier detection in large multidimensional data sets. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*. ACM Press, Madison, Wisconsin.

Fan, H., Zaïane, O. R., Foss, A., and Wu, J. 2006. A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. In *PAKDD*, W. K. Ng, M. Kitsuregawa, J. Li, and K. Chang, Eds. Lecture Notes in Computer Science, vol. 3918. Springer, 557–566.

Ghoting, A., Otey, M. E., and Parthasarathy, S. 2004. Loaded: Link-based outlier and anomaly detection in evolving data sets. In *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, 387–390.

Hand, D. J. and Till, R. J. 2001. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning 45,* 2, 171–186.

He, Z., Deng, S., and Xu, X. 2005. A unified subspace outlier ensemble framework for outlier detection. In *WAIM*, W. Fan, Z. Wu, and J. Yang, Eds. Lecture Notes in Computer Science, vol. 3739. Springer, 632–637.

He, Z., Xu, X., and Deng, S. 2003. Discovering cluster-based local outliers. *Pattern Recogn. Lett. 24,* 9-10, 1641–1650.

Joanes, D. N. and Gill, C. A. 1998. Comparing measures of sample skewness and kurtosis. *Journal of the Royal Statistical Society (Series D): The Statistician 47,* 1, 183–189.

Knorr, E. M. and Ng, R. T. 1998. Algorithms for mining distance-based outliers in large datasets. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*. Morgan Kaufmann, San Francisco, CA, USA, 392–403.

Knorr, E. M., Ng, R. T., and Tucakov, V. 2000. Distance-based outliers: algorithms and applications. *The VLDB Journal 8,* 3-4, 237–253.

Knuth, D. E. 2006. *Art of Computer Programming, Volume 4, Fascicle 4: Generating All Trees.* Addison-Wesley Professional.

Lazarevic, A. and Kumar, V. 2005. Feature bagging for outlier detection. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.* ACM, New York, NY, USA, 157–166.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. 2008a. Isolation Forest. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining.* IEEE Computer Society, 413–422.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. 2008b. Spectrum of variable-random trees. *Journal of Artificial Intelligence Research 32*, 355–384.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. 2010a. Can isolation-based anomaly detectors handle arbitrary multi-modal patterns in data? Tech. Rep. TR2010/2, Monash University.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. 2010b. On detecting clustered anomalies using SCiForest. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.* Barcelona, Spain.

Liu, R. Y., Parelius, J. M., and Singh, K. 1999. Multivariate analysis by data depth: Descriptive statistics, graphics and inference. *The Annals of Statistics 27,* 3 (Jun), 783–840.

Murphy, R. B. 1951. On tests for outlying observations. Ph.D. thesis, Princeton University.

Papadimitriou, S., Kitagawa, H., Gibbons, P., and Faloutsos, C. 2003. Loci: fast outlier detection using the local correlation integral. *Data Engineering, 2003. Proceedings. 19th International Conference on*, 315–326.

Preiss, B. R. 1999. *Data Structures and Algorithms with Object-Oriented Design Patterns in Java.* Wiley.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*, 1 ed. Morgan Kaufmann.

Ramaswamy, S., Rastogi, R., and Shim, K. 2000. Efficient algorithms for mining outliers from large data sets. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data.* ACM Press, New York, NY, USA, 427–438.

Rocke, D. M. and Woodruff, D. L. 1996. Identification of outliers in multivariate data. *Journal of the American Statistical Association 91,* 435, 1047–1061.

Rousseeuw, P. J. and Leroy, A. M. 1987. *Robust regression and outlier detection.* John Wiley & Sons, Inc., New York, NY, USA.

Ruskey, F. 1980. On the average shape of binary trees. *SIAM Journal on Algebraic and Discrete Methods 1,* 1, 43–50.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation 13,* 7, 1443–1471.

Shi, T. and Horvath, S. 2006. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics 15,* 1 (March), 118–138.

Sing, T., Sander, O., Beerenwinkel, N., and Lengauer, T. 2005. ROCR: visualizing classifier performance in r. *Bioinformatics 21,* 20, 3940–3941.

Song, X., Wu, M., Jermaine, C., and Ranka, S. 2007. Conditional anomaly detection. *IEEE Trans. on Knowl. and Data Eng. 19,* 5, 631–645.

Tan, P.-N., Steinbach, M., and Kumar, V. 2005. *Introduction to Data Mining.* Addison-Wesley.

Tang, J., Chen, Z., Fu, A. W.-C., and Cheung, D. W.-L. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *PAKDD '02: Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining.* Springer-Verlag, London, UK, 535–548.

Tao, Y., Xiao, X., and Zhou, S. 2006. Mining distance-based outliers from large databases in any metric space. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, New York, NY, USA, 394–403.

Tax, D. M. J. and Duin, R. P. W. 2004. Support vector data description. *Machine Learning 54,* 1, 45–66.

Tukey, J. W. 1977. *Exploratory Data Analysis.* Addison-Wesley.

Williams, G., Baxter, R., He, H., Hawkins, S., and Gu, L. 2002. A comparative study of rnn for outlier detection in data mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining.* IEEE Computer Society, Washington, DC, USA, 709–712.

Wu, M. and Jermaine, C. 2006. Outlier detection by sampling with accuracy guarantees. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, New York, NY, USA, 767–772.

Yamanishi, K., Takeuchi, J.-I., Williams, G., and Milne, P. 2000. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, 320–324.

Yu, X., Tang, L. A., and Han, J. 2009. Filtering and refinement: A two-stage approach for efficient and effective anomaly detection. In *ICDM '09: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining.* IEEE Computer Society, Washington, DC, USA, 617–626.

APPENDIX

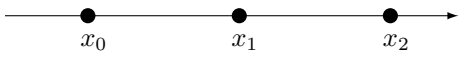
A. A PROBABILISTIC EXPLANATION TO ISOLATION TREES

$P(.)$	The probability function
$E(.)$	The expectation function
$c(n)$	average height of random binary trees,$c(n) = 2H(n-1) - 2(n-1)/n$
$H(i)$	harmonic number $H(i) \approx \ln(i) + 0.5772156649$ (Euler's constant)
C_p	The Catalan Number, $C_p = \binom{2p}{p} - \binom{2p}{p-1}$
C_{pr}	The Catalan Number, $C_{pr} = \binom{p+r}{p} - \binom{p+r}{p-1}$

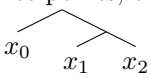
Table VII. Symbols and Notations

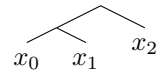
In this appendix, in order to understand the susceptibility of isolation in terms of the average path length, we generalize the expected path length to a summation of a series from all possible isolation trees for univariate cases. Symbols and notations used in this appendix can be found in Table VII.

Let us start with a simple model. Given a simple univariate distribution of $\{x_0, x_1, x_2\} \in X$, where $x_0 < x_1 < x_2$, as shown below.



Given these three points, there are two possible tree structures:

(Tree A) 

(Tree B) 

In this case, if the initial split point falls between x_0 and x_1, then tree A is constructed, else tree B. By the Law of Large Numbers, the expected path length for each point can be calculated exactly according to the possible tree structures above:

$$E(h(x_0)) = P(h(x_0) = 1) \times 1 + P(h(x_0) = 2) \times 2,$$
$$E(h(x_1)) = P(h(x_1) = 1) \times 1 + P(h(x_1) = 2) \times 2,$$
$$E(h(x_2)) = P(h(x_2) = 1) \times 1 + P(h(x_2) = 2) \times 2.$$

In this simple case, assuming uniform distribution, to calculate the probability of each path length component, we denote $D_{0,1}$ as the distance between x_0 and x_1. In this simple example, the first split point of a tree basically decides which possible structure will be formed. So, the probability of $P(h(x_0) = 1) = P(h(x_2) = 2) = P(A) = \frac{D_{0,1}}{D_{0,2}}$ and $P(h(x_0) = 2) = P(h(x_2) = 1) = P(B) = \frac{D_{1,2}}{D_{0,2}}$.

In general, where $|X| > 1$, the number of possible trees is C_j, a Catalan number of j, where $j = |X| - 1$. The number grows as follows: $\{1, 2, 5, 14, 42, 132, 429, 1430, 4862\}$ for $|X| \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Note that in cases where the number of samples is more than three, each probabilistic component is mapped to multiple possible tree structures. For each data point of interest

x, the expected path length $E(h(x))$ is a summation of a series of the possible path lengths with probabilistic components such that:

$$E(h(x)) = \sum_{l} P(h(x) = l) \times l. \tag{3}$$

The generalised possible path lengths for any fringe points are $h(x) \in [1, |X| - 1]$; and the generalised possible path lengths for any non-fringe points are $h(x) \in [2, |X| - 1]$. The sum of the probabilistic components of all possible path lengths $\sum_l P(h(x) = l) = 1$.

Assuming that each possible tree structure is equally probable, i.e., uniform distribution; the term $P(h(x) = l)$ can be estimated by $\frac{t_{lmj}}{C_j}$ [Knuth 2006], where t_{lmj} is the total number of possible trees that has $h(x_m) = l$ with j internal nodes,

$$t_{lmj} = \sum_{u=0}^{n} \left(\begin{array}{c} l \\ u \end{array} \right) C_{(m-u)(m-l)} C_{(j-m-l+u)(n-m-l)}. \tag{4}$$

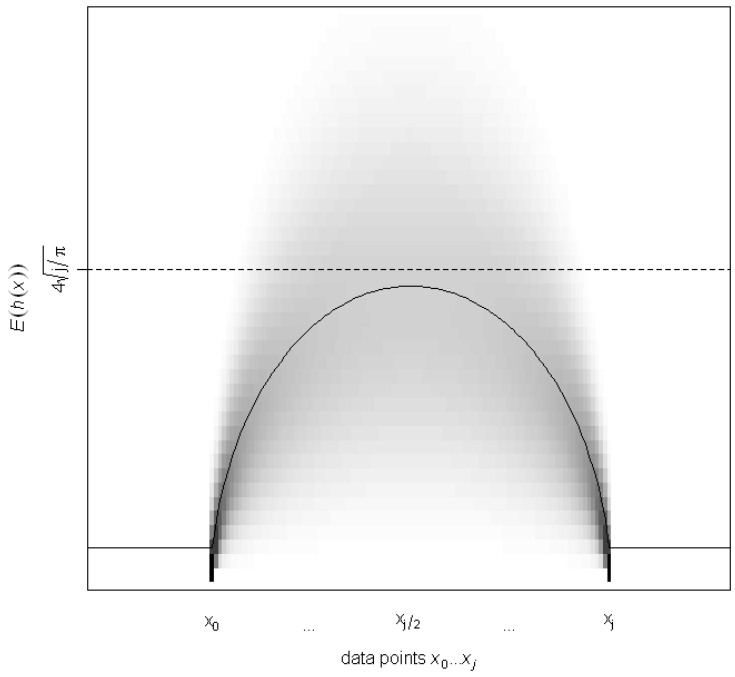


Fig. 16. The average path length $E(h(x))$ of randomly generated binary trees for uniformly distributed data points. $P(h(x))$ is represented by grey scale distribution.

The average path length of a data point x_m is $E(h(x)) = \frac{h_{mj}}{C_j}$[Knuth 2006], where

h_{mj} is the sum of path lengths for x_m in all possible trees with j internal nodes,

$$h_{mj} = 2 \left(\begin{array}{c} 2m \\ m \end{array} \right) \left(\begin{array}{c} 2j-2m \\ j-m \end{array} \right) \frac{(2m+1)(2j-2m+1)}{(j+1)(j+2)} - C_j, \tag{5}$$

and $m \in \{0, ..., j\}$.

The average path length of all possible trees[5] was initially investigated by [Ruskey 1980], and it is shown in Figure 16, which has a **dome** shape. The probabilistic component $P(h(x) = l)$ in Equation 3 gives the grey scale distribution in Figure 16. The height of this shape is approximately $4\sqrt{j/\pi}$ [Knuth 2006]. For regions that have no data point, i.e., $x < x_0$ and $x > x_j$, those regions share the same path length as their closest fringe points. In Figure 16, the dome shape reveals that the fringe points have much lower expected path lengths as compared to those of the core points.

[5]Ruskey called this the shape of random trees.

B. IFOREST'S PERFORMANCE AGAINST VARIOUS NUMBER OF TREES

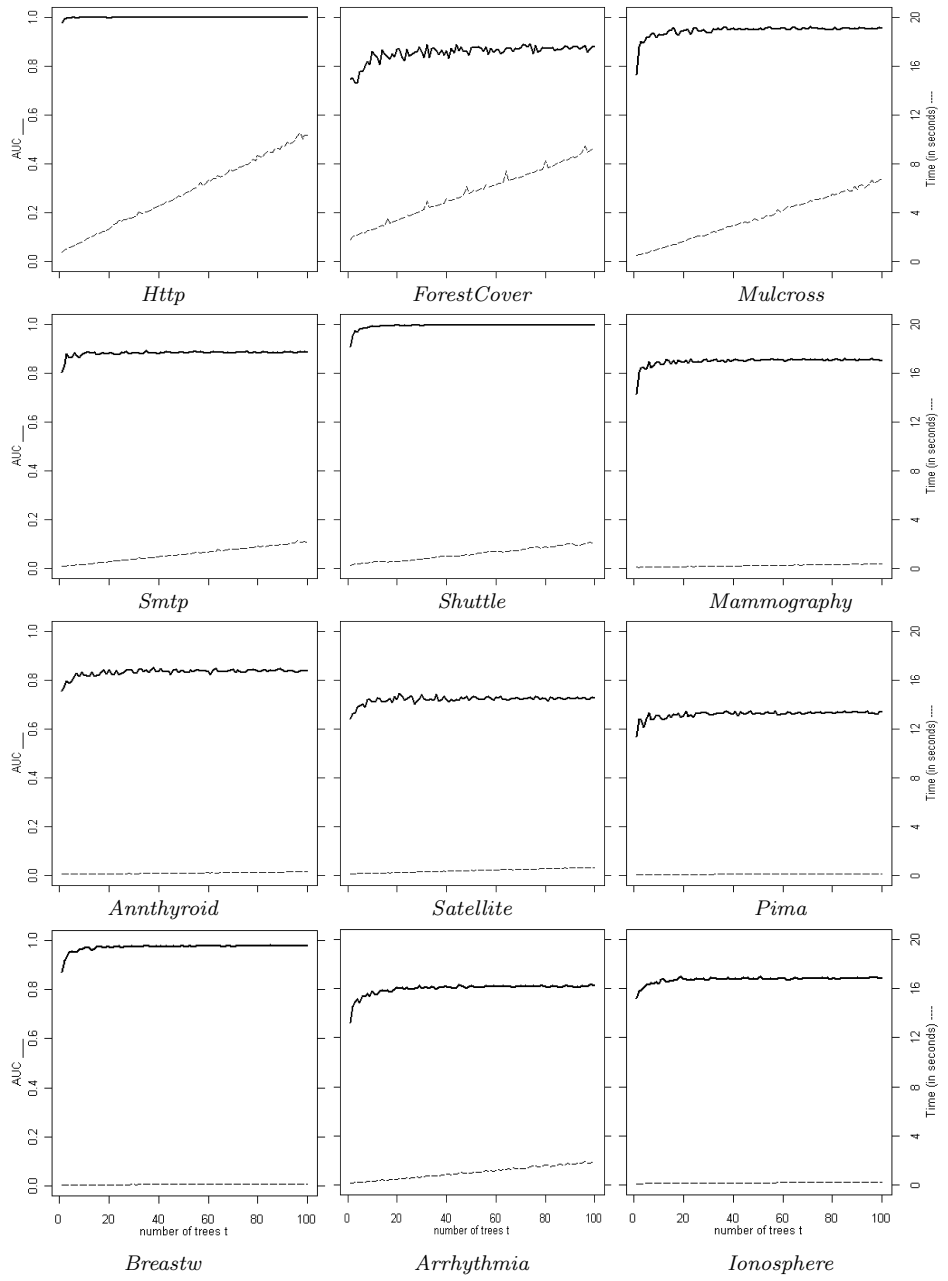


Fig. 17. *i*Forest's AUC performance (y1-axis) and processing time (y2-axis) versus the different number of trees t used (x-axis).

C. IFOREST'S PERFORMANCE AGAINST VARIOUS SAMPLING SIZE

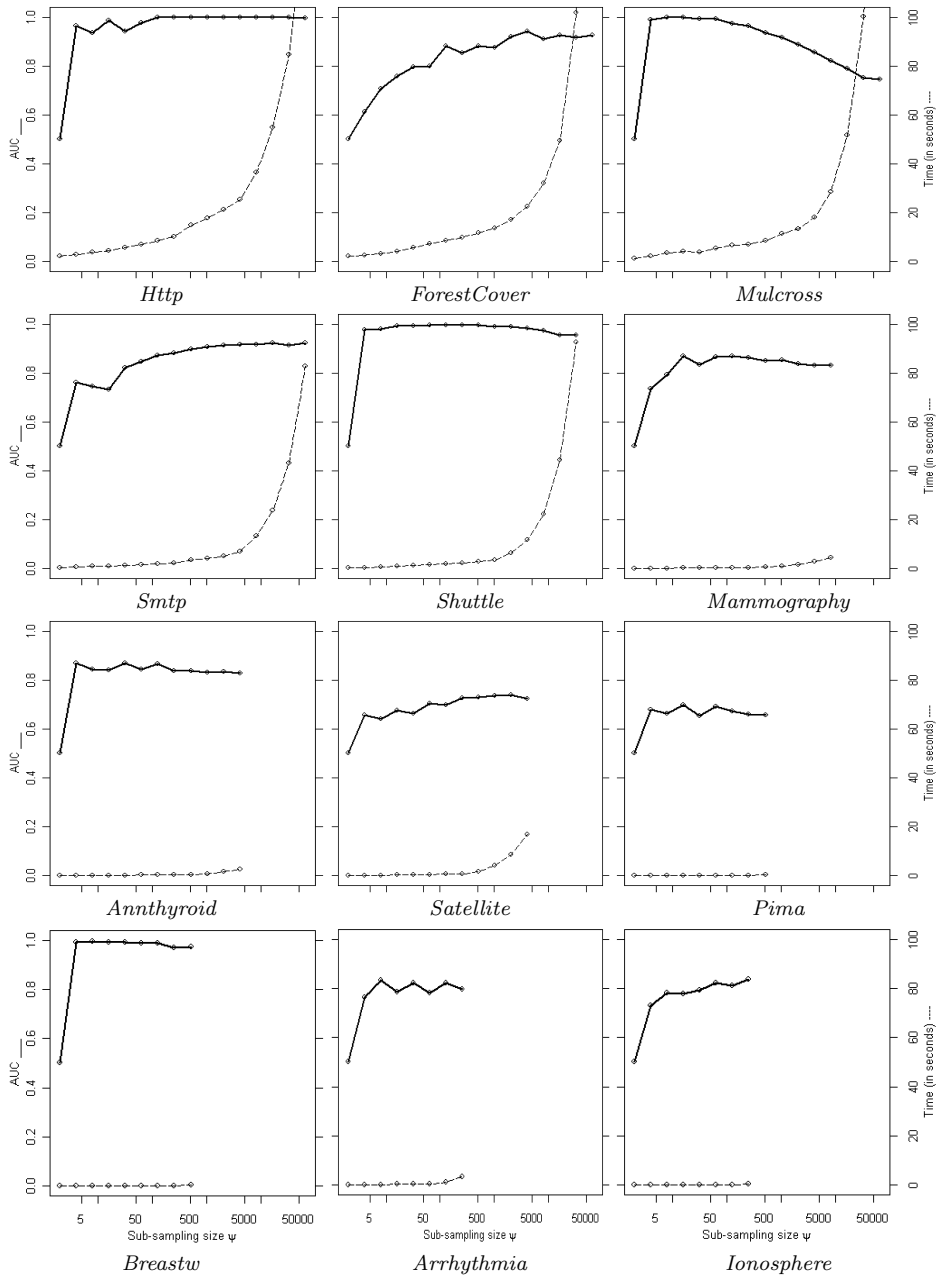


Fig. 18. *i*Forest's AUC performance (y1-axis) and processing time (y2-axis) versus the different subsampling sizes ψ in log scale (x-axis).

D. IFOREST'S PERFORMANCE USING KURTOSIS AS FEATURE SELECTOR

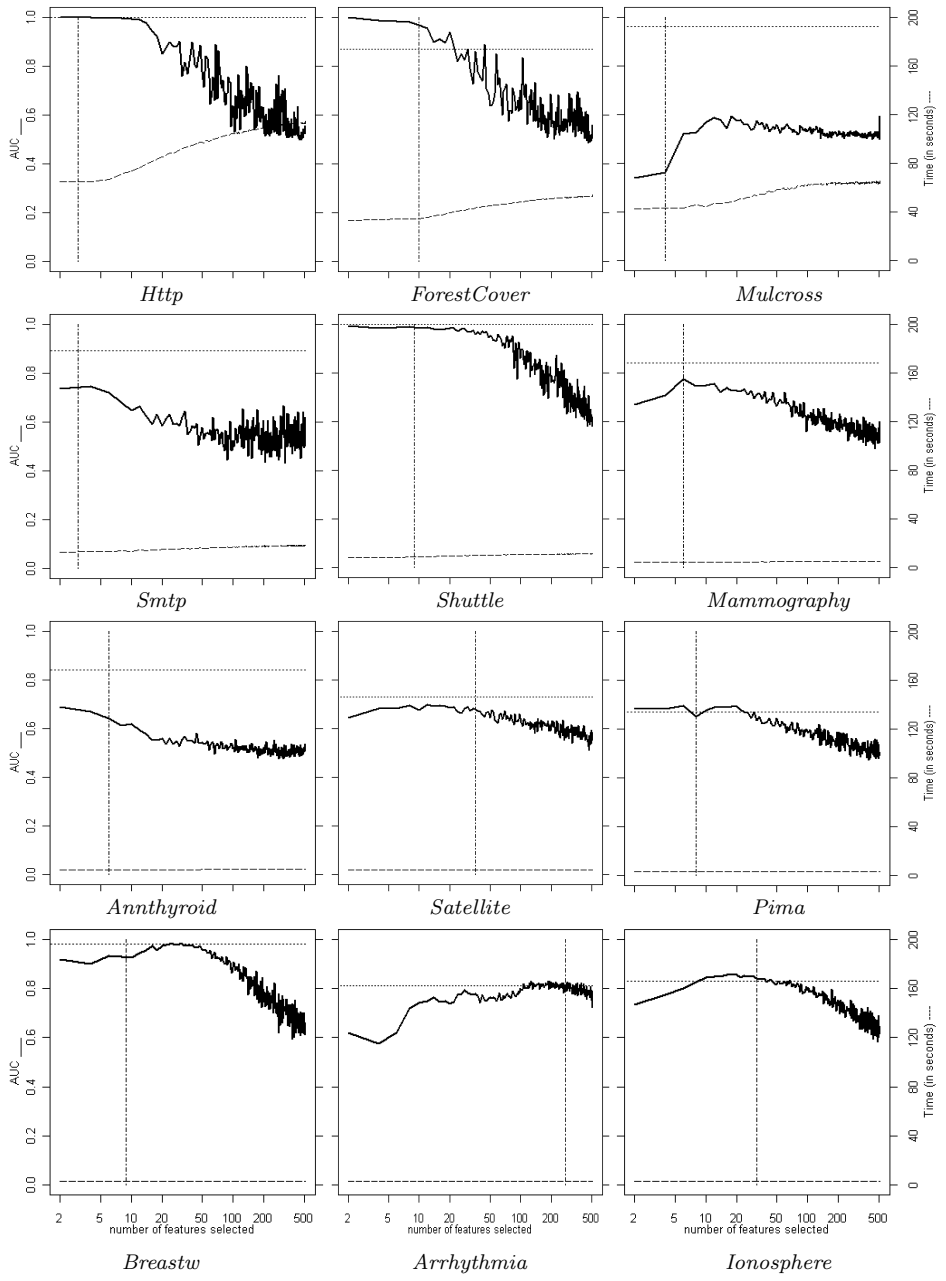


Fig. 19. *i*Forest's AUC performance (y1-axis, solid lines) and processing time (y2-axis, dashed lines) versus the different number of attributes selected from Kurtosis (x-axis, log scale) in irrelevant attribute added data. Dotted lines denote the AUC performance of *i*Forest using the original data without added attributes. Dotted dash lines indicate the original number of attributes.

E. AUC CALCULATION FOR ANOMALY DETECTION

AUC is known as the area under receiver operating characteristic curve. In data mining, it is usually used to measure the overall performance of classifiers regardless of the threshold between the true positives and true negatives. In the context of anomaly detection, anomalies are treated as the positive class. To calculate AUC, a simple approach adopted from [Hand and Till 2001] is as follows:

Algorithm 4 : AUC

1: let n_a be the number of true anomalies.
2: let n_n be the number of true normal points.
3: rank all instances according to their anomaly scores in descending order.
4: let S be the sum of rankings of the actual anomalies, $S = \sum_{i=1}^{n_a} r_i$, where r_i is the rank of the i^{th} anomaly in the ranked list.
5: $AUC = \dfrac{S - (n_a^2 + n_a)/2}{n_a n_n}$