Isolation Forest

Fei Tony Liu, Zhi-Hua Zhou

Presenter: Jaehyeong Ahn

Department of Applied Statistics, Konkuk University

jayahn0104@gmail.com

Contents

Introduction

- Anomalies are data patterns that have different data characteristics from normal instances
- The detection of anomalies has significant relevance and often provides critical actionable information in various application domains (e.g. fraudulent use of credit cards, unusual computer network traffic pattern, $\dots$)
- Most existing model-based approaches to anomaly detection construct a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies
- However there are two major drawbacks of this approach
  1. The anomaly detector is optimized to profile normal instances, but not optimized to detect anomalies which causes too many false alarms
  2. Many existing methods are constrained to low dimensional data and small data size because of their high computational complexity

Introduction

In this work,

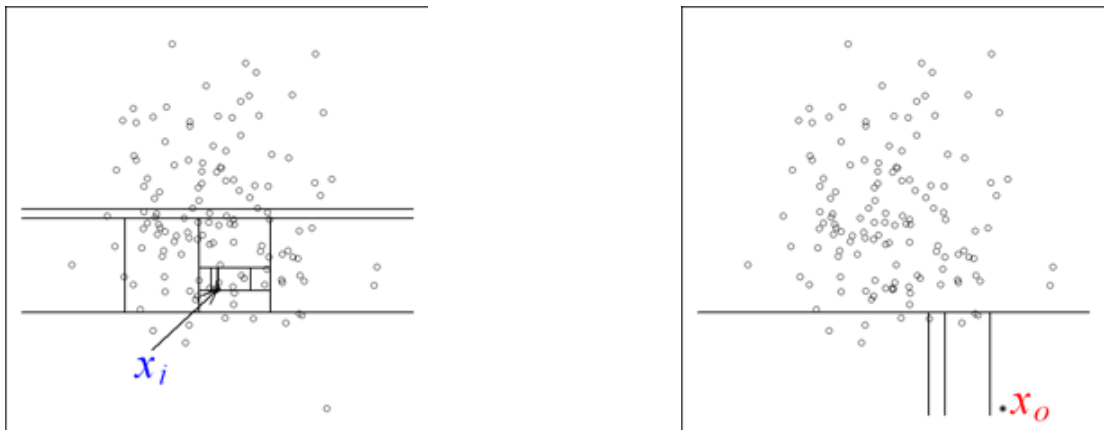
- We introduce the method called "Isolation Forest" or "iForest" which is suggested by [Liu et.al.]
- iForest proposes a different type of model-based method that explicitly isolates anomalies rather than profiles normal instances
  - iForest utilizes no distance or density measures to detect anomalies. This eliminates major computational cost of distance calculation
  - iForest has a linear time complexity with a low constant and a low memory requirement
  - iForest has the capacity to scale up to handle extremely large data size and high-dimensional problems with a large number of irrelevant attributes
- Assume:
  1. Anomalies are the minority consisting of fewer instances
  2. Anomalies have attribute-values that are very different from those of normal instances

Basic Idea

- Construct an unsupervised decision tree which randomly partition the sample space with binary split until every single instance is isolated. Call this tree as **iTree**
- Because of their susceptibility to isolation, anomalies will be isolated closer to the root of the tree; whereas normal points will be isolated at the deeper end of the tree
- Build an **iForest** which is an ensemble of iTrees for sub-sampled, bootstrapped without replacement, data sets
- Calculate the average path length of each data point and get anomaly score by using some metric
- Identify data points as anomalies which have high anomaly score

Basic Idea

- Anomalies are more susceptible to isolation and hence have short path lengths
  - x_i is a normal point and x_o is an anomaly



(a) Isolating x_i (b) Isolating x_o

- x_i requires 12 random partitions to be isolated
- Whereas x_o requires only 4 partitions to be isolated

$\Rightarrow$ the number of partitions required to isolate a normal point is greater than anomaly

(the number of partitions is equivalent to the path length from the root node to a terminal node in tree)

Basic Idea

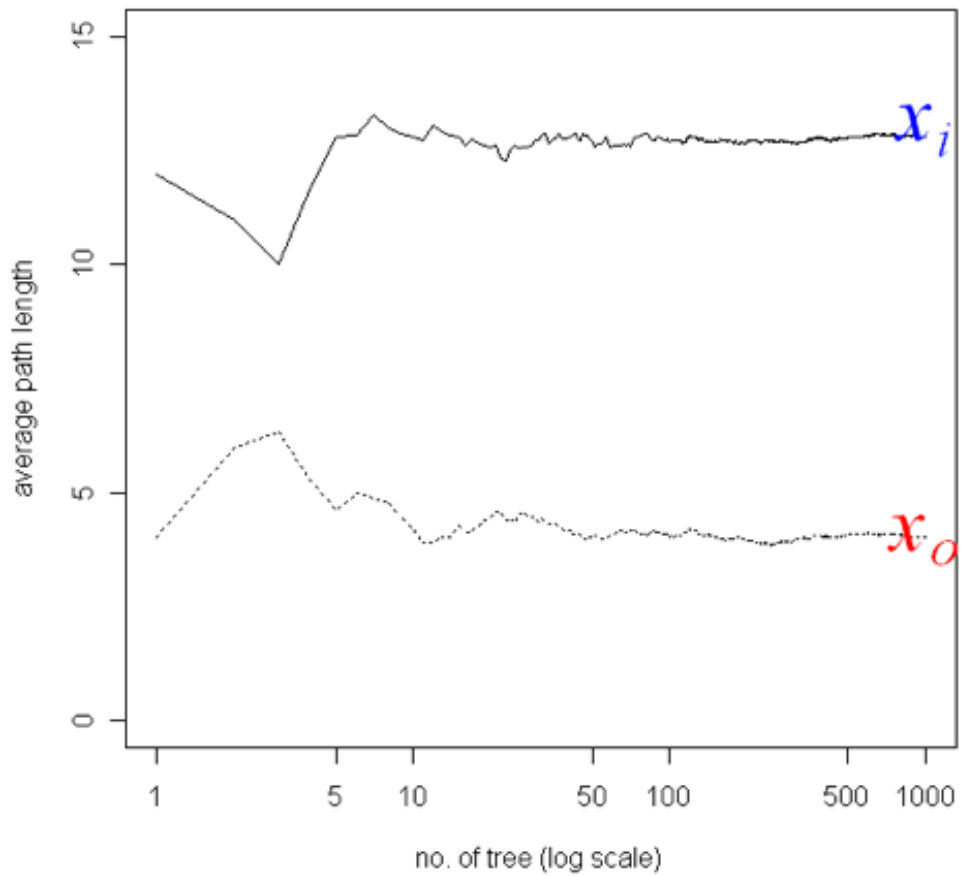


Figure: Average path lengths converge

Notation

- $X = \{x_1, \cdots, x_n\}, \quad x_i \in \mathbb{R}^d$ for all $i = 1, \cdots, N$
  - (only consider continuous valued attributes)
- T is a node of an isolation tree
  - T is either an external-node (terminal-node) with no child, or an internal-node with one test and exactly two daughter nodes (T_l, T_r)
  - A test consists of an attribute q and a split value p such that the test $q < p$ divides data points into T_l and T_r
- $h(x)$: Path Length of a point x which is measured by the number of edges x traverses an iTree from the root node until the traversal is terminated at an external node
- $c(n)$: Average path length of unsuccessful search in BST(Binary Search Tree)[B. R. Preiss,1999]. This is used for normalization of $h(x)$, $(H(i) \approx ln(i) + 0.5772156649)$

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{1}$$

- $s(x, n)$: Anomaly score

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

- $s(x, n)$: Anomaly score

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

  - when $E(h(x)) \to c(n), \quad s \to 0.5$
  - when $E(h(x)) \to 0, \quad s \to 1$
  - when $E(h(x)) \to n - 1, \quad s \to 0$

$$\Rightarrow s \text{ is monotonic to } h(x)$$

- $0 < s \leq 1$ for $0 < h(x) \leq n - 1$
  - (a) if instances return s very close to 1, then they are definitely anomalies,
  - (b) if instances have s much smaller than 0.5, then they are quite safe to be regarded as normal instances,
  - (c) if all the instances return $s \approx 0.5$, then the entire sample does not really have any distinct anomaly

Methodology

- Anomaly detection using iForest is a two-stage process
- The first (training) stage builds isolation trees using sub-samples, bootstrapped sample without replacement, of the training set
- The second (testing) stage passes the test instances through isolation trees to obtain an anomaly score for each instance

Training Stage

- In the training stage, iTrees are constructed by recursively partitioning the given training set until instances are isolated or a specific tree height is reached
- The tree height limit l is automatically set by the sub-sampling size $\psi : l = ceiling(\log_2 \psi)$ which is approximately the average tree height [D. E. Knorr, 1998]
- The reason of setting a tree height limit is that we are only interested in data points which have shorter than average path lengths, as those points are more likely to be anomalies
- Details of the training stage can be found in following Algorithms 1 and 2
- The complexity of the training an iForest is $O(t\psi \log \psi)$

Training Stage

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

Inputs : X - input data, t - number of trees, ψ - sub-sampling size
Outputs: a set of t $iTrees$
begin
    set height limit $l = ceiling(\log_2 \psi)$;
    for $i = 1$ *to* t **do**
        $X^{'} \leftarrow sample(X, \psi)$;
        $Forest \leftarrow Forest \cup iTree(X^{'}, 0, l)$;
    end
    return $Forest$
end

Algorithm 2: $iTree(X, e, l)$

Inputs : X - input data, e - current tree height, l - height limit
Outputs: an iTree
begin
  if $e \geq l$ *or* $|X| \leq 1$ **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 from max and min values of attribute q in X
    $X_l \leftarrow filter(X, q < p)$
    $X_r \leftarrow filter(X, q \leq p)$

    return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
      $Right \leftarrow iTree(X_r, e+1, l),$
      $SplitAtt \leftarrow q,$
      $SplitValue \leftarrow p\}$
  end
end

Evaluating Stage

- In the evaluating stage, an anomaly score s is derived from the expected path length $E(h(x))$ for each test instances
- $E(h(x))$ are derived by passing instances through each iTree in an iForest
- Using $PathLength$ (Algorithm3) function, a single path length $h(x)$ is derived by counting the number of edges e from the root node to a terminating node
- When $h(x)$ is obtained for each tree of the ensemble, an anomaly score is produced by computing $s(x, \psi)$ in Equation 2
- To find the top m anomalies, simply sorts the data using s in descending order. The first m instances are the top m anomalies
- The complexity of the evaluation process is $O(nt \log \psi)$, where n is the testing data size

Evaluating Stage

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

Inputs : x - an instance, T - an iTree, e - current path length;
to be initialized to zero when first called
Outputs: path length of x
begin
  if T *is an external node* **then**
    return $e + c(T.size)\{c(.)$is defined in Equation 1$\}$
  end
  $a \leftarrow T.splitAtt$;
  if $x_a < T.splitValue$ **then**
    return $PathLength(x, T.left, e + 1)$;
  else if $\{x_a \geq T.splitValue\}$ **then**
    return $PathLength(x, T.right, e + 1)$
  end
end

Empirical Experiments

- Given data sets
  - It is assumed that anomaly lables are unavailable in the training stage
  - Anomaly lables are available in the evaluation stage to compute the performance measure AUC

	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%)

Figure: Table of Data properties

Empirical Experiments

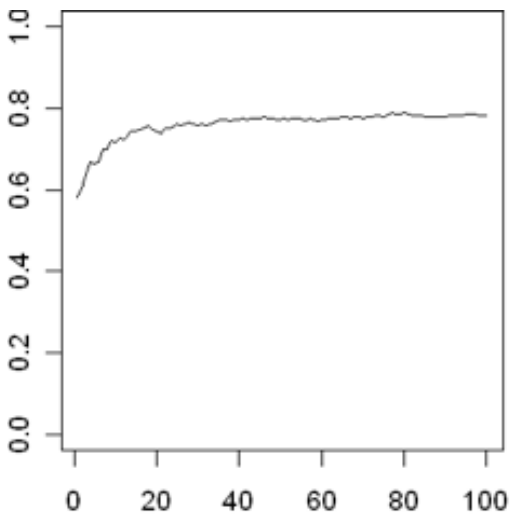
- Comparison with ORCA [Bay, 2003], LOF [Knox, 1998] and Random Forests [Shi, 2006]
  - For all the experiments, actual CPU time and Area Under Curve (AUC) on ROC curve are reported as measurements

	AUC				Time (seconds)					
	iForest	ORCA	LOF	RF	iForest			ORCA	LOF	RF
					Train	Eval.	Total			
Http (KDDCUP99)	**1.00**	0.36	NA	NA	0.25	15.33	**15.58**	9487.47	NA	NA
ForestCover	**0.88**	0.83	NA	NA	0.76	15.57	**16.33**	6995.17	NA	NA
Mulcross	**0.97**	0.33	NA	NA	0.26	12.26	**12.52**	2512.20	NA	NA
Smtp (KDDCUP99)	**0.88**	0.80	NA	NA	0.14	2.58	**2.72**	267.45	NA	NA
Shuttle	**1.00**	0.60	0.55	NA	0.30	2.83	**3.13**	156.66	7489.74	NA
Mammography	**0.86**	0.77	0.67	NA	0.16	0.50	**0.66**	4.49	14647.00	NA
Annthyroid	**0.82**	0.68	0.72	NA	0.15	0.36	**0.51**	2.32	72.02	NA
Satellite	**0.71**	0.65	0.52	NA	0.46	1.17	**1.63**	8.51	217.39	NA
Pima	0.67	**0.71**	0.49	0.65	0.17	0.11	0.28	**0.06**	1.14	4.98
Breastw	**0.99**	0.98	0.37	0.97	0.17	0.11	0.28	**0.04**	1.77	3.10
Arrhythmia	**0.80**	0.78	0.73	0.60	2.12	0.86	2.98	**0.49**	6.35	2.32
Ionosphere	0.85	**0.92**	0.89	0.85	0.33	0.15	0.48	**0.04**	0.64	0.83

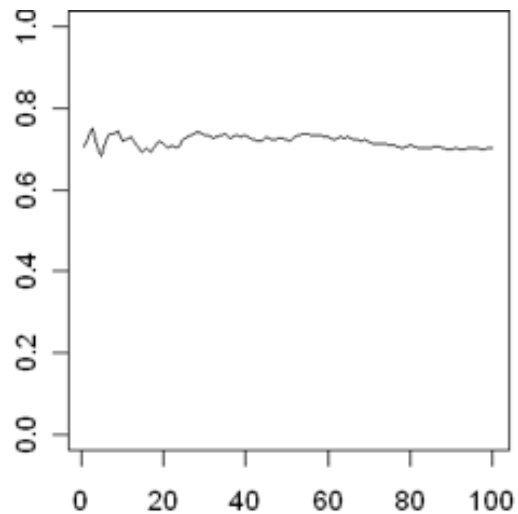
Figure: Table of Models

Empirical Experiments

- Early Convergence of AUC on iForest
  - The performance of iForest is stable in wide range of t
  - Since increasing t also increases processing time, the early convergence of AUC suggests that iForest's execution time can be further reduces if t is tuned to a data set



(a) Arrhythmia



(b) Satellite

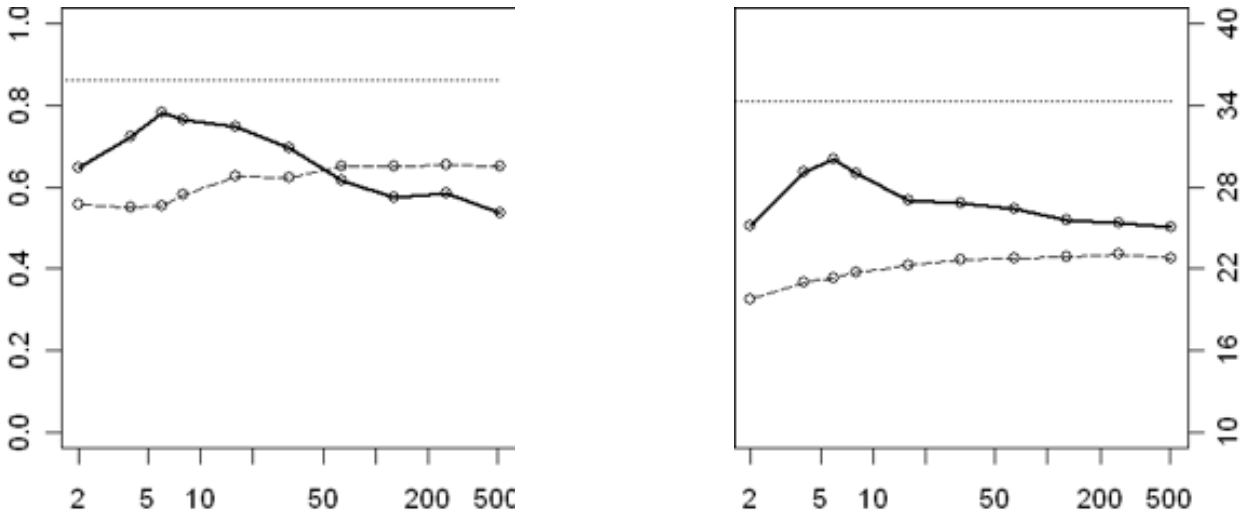
Figure: Detection performance AUC (y-axis) is converged at a small t (x-axis)

Empirical Experiments

- High Dimensional Data
  - One of the important challenges in anomaly detection is high dimensional data
  - In this experiment, we study a special case in which high dimensional data sets has a large number of irrelevant attributes or background noises
  - Used data sets: *Mammorgraphy* (n=11183, d=6), *Annthyroid* (n=6832, d=6)
  - For each data set, 506 random variables added to simulate background noise. Thus there is a total of 512 attributes in each data sets
  - We use a simple statistical test, Kurtosis, to select an attribute subspace from the sub-sample before constructing each iTree
  - After Kurtosis has provided a ranking for each attribute, a subspace of attributes is selected according to this ranking to construct each tree

Empirical Experiments

- High Dimensional Data



(a) *Mammography* (b) *Annthyroid*

Figure: (left y-axis, solid lines): AUC, (right y-axis, dashed lines): Processing time in seconds, (x-axis): subspace size

- Figure shows that processing time (dashed line) remains less than 30 seconds for the whole range of subspace size
- AUC (solid line) peaks when subspace size is the same as the number of original attributes only (at 6)

Summary

- In this work, we introduced a fundamentally different model-based method that focuses on anomaly isolation rather than normal instance profiling
- Taking advantage of anomalies' nature of 'few and different', iTree isolates anomalies closer to the root of the tree as compared to normal points
- This unique characteristic allows iForest to build partial models and employ only a tiny proportion of training data to build effective models
- As a result, iForest has a linear time complexity with a low constant and a low memory requiremnt which is ideal for high volume data sets
- Empirical experiments show that iForest out performs ORCA, LOF and RF in terms of AUC and execution time, especially in large data sets
- Next works:
  - Deal with categorical data
  - Develop an elaborated random partition method
  - Better dimension reduction (or variable selection) method

References

1 Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." In 2008 Eighth IEEE International Conference on Data Mining, pp. 413-422. IEEE, 2008.

2 Preiss, Bruno R. "Data Structures and Algorithms." (1999).

3 Knuth, Donald E. "The Art of Computer Programming, Volume 3: Searching and Sorting." Addison-Westley Publishing Company: Reading, MA (1973).

4 Bay, Stephen D., and Mark Schwabacher. "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, pp. 29-38. 2003.

5 Knox, Edwin M., and Raymond T. Ng. "Algorithms for mining distancebased outliers in large datasets." In Proceedings of the international conference on very large data bases, pp. 392-403. Citeseer, 1998.

6 Shi, Tao, and Steve Horvath. "Unsupervised learning with random forest predictors." Journal of Computational and Graphical Statistics 15, no. 1 (2006): 118-138.