AI in Anomaly Detection and Image Compression

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Kaushal Kishore
(111601008)

under the guidance of

Dr. Chandra Shekar

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**AI in Anomaly Detection and Image Compression**" is a bonafide work of **Kaushal Kishore** (**Roll No. 111601008**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my supervision and that it has not been submitted elsewhere for a degree.*

Dr. Chandra Shekar

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to show my greatest appreciation to Dr. Chandra Shekar. I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. Without his encouragement and guidance, this project would not have materialized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter discusses anomaly detection, its use cases and some major challenges.

1.1 Anomaly Detection

Anomaly detection (also outlier detection) is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies also known as outliers, novelties, noise, deviations and exceptions.

Three broad categories of anomaly detection techniques exist:

1. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set.

2. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to

many other statistical classification problems is the inherently unbalanced nature of outlier detection).

3. Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then test the likelihood of a test instance to be generated by the learnt model.

We will restrict ourselves to unsupervised anomaly detection and semi-supervised anomaly detection problem.

1.2 Use Cases

The ability to detect anomalies has significant relevance, and anomalies often provides critical and actionable information in various application domains.

Identification of potential outliers is important for the following reasons: [1]

1. An outlier may indicate bad data. For example, the data may have been coded incorrectly, or an experiment may not have been run correctly. If it can be determined that an outlying point is in fact erroneous, then the outlying value should be deleted from the analysis (or corrected if possible).

2. In some cases, it may not be possible to determine if an outlying point is bad data. Outliers may be due to random variation or may indicate something scientifically interesting. In any event, we typically do not want to simply delete the outlying observation.

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 unauthorised access. These applications demand anomaly detection algorithms with high detection accuracy and fast execution.

1.3 Masking and Swamping

Masking and swamping is the biggest problem affecting any anomaly detection algorithm.

Masking is the existence of too many anomalies concealing their own presence. It happens when anomaly clusters become large and dense. For example, if we are testing for a single outlier when there are in fact more outliers, these additional outliers may influence the value of the test statistic enough so that no points are declared as outliers.

On the other hand, 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. For example, if we are testing for two or more outliers when there is in fact only a single outlier, both points may be declared outliers.

Masking is one reason that trying to apply a single outlier test sequentially can fail. For example, if there are multiple outliers, masking may cause the outlier test for the first outlier to return a conclusion of no outliers. So the testing is not performed for any additional outliers.

1.4 Concept drift

In the case of streaming data, the anomaly context can change over time. For example, consider a user's behaviour change from one system to another. The anomaly detection algorithm should adapt to this change in the behaviour of the external agent. This deviation of the normal behaviour time to time is called concept drift. Any online anomaly detection algorithm must have a way to deal with this.

1.5 Organization of The Report

This chapter [1] provides a background for the topics covered in this report. We provided a description of anomaly detection problem and discussed some use cases. Then we discussed some challenges to anomaly detection problem: masking, swamping and concept drift. In

the next chapter [2] we will discuss a very efficient ensemble method Isolation Forest for anomaly detection. In chapter [3] we will discuss another ensemble method PIDForest which has been recently developed. The major drawback of the above mentioned algorithms is that they are used in offline setting without dealing with concept drift. Most of the anomaly detection algorithm is offline and fail to address the problem of concept drift. In chapter [4] we will present some methods to address these issues. In chapter [5] we will review our work did till mid-sem. And finally in chapter [6], we conclude with some future works.

Code Repository: `https://github.com/KishoreKaushal/AnomalyDetection`

Report Repository: `https://github.com/KishoreKaushal/btp-report`

Chapter 2

Isolation Forest

2.1 Isolation based anomaly detection

Isolation is the process or fact of isolating or being isolated. The authors of [2] proposed an isolation based anomaly detection which takes advantage of two quantitative properties of anomalies:

1. They are the minority consisting of few instances.

2. They have attribute-values that are very different from those of normal instances.

Hence, 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. Lui et al. [2] proposed 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.

The proposed method, called Isolation Forest (iForest) builds an ensemble 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.

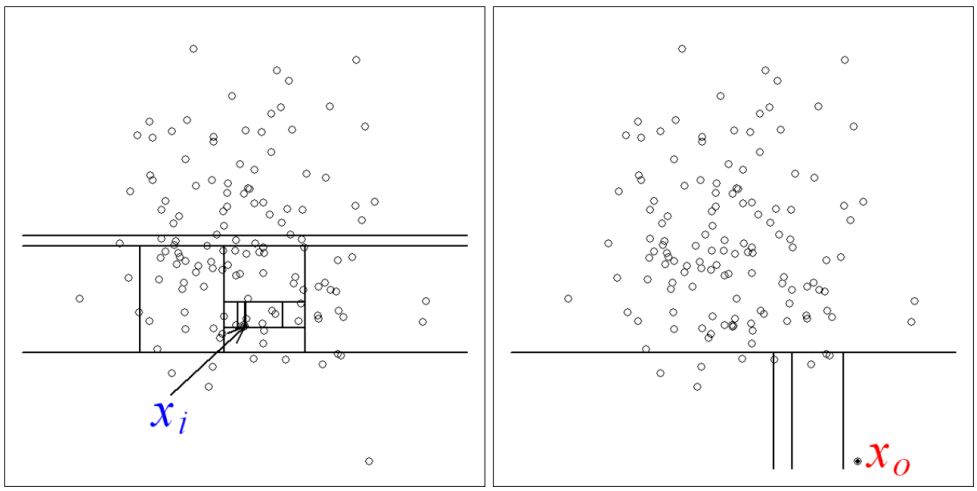


Fig. 2.1: **Left**: a normal point x_i requires twelve random partitions to be isolated; **Right**: an anomaly x_o requires only four partitions to be isolated.
Source: Lui et al. [2]

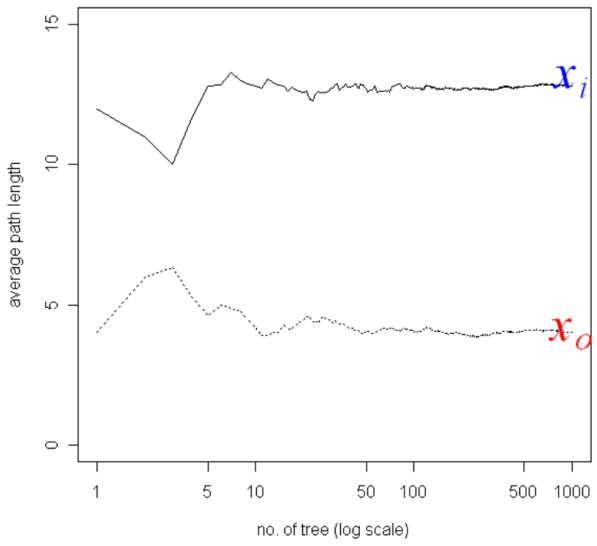


Fig. 2.2: Averaged path lengths of x_i and x_o converge when the number of trees increases.
Source: Lui et al. [2]

2.2 Training Stage

Formally, isolation tree is defined as follows:

Definition: 1 *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 instances $X' \subset X$ is used to build an isolation tree[2]. We recursively divide X' by randomly selecting an attribute q and a split value p, until either: i) Node has only one instance ii) Or, all data at the node have the same values.*

Definition: 2 *Isolation forest is defined as 4-tuple (X, t, ψ, S) where*

- *X is input data,*
- *t is number of trees,*
- *ψ is subsampling size and*
- *S is the set of isolation trees.*

The elements of set S is constructed[1] by sampling ψ instances from X without replacement.

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

Complexity: Time - $O(t\psi^2)$, Space - $O(t\psi)$
Input: X - input data, t - number of trees, ψ - subsampling size
Output: List of $iTrees$

```
Forest ← EmptyList
for i = 1 to t do
    X' ← sampleWithoutReplacement(X, ψ)
    Forest ← Forest ∪ iTree(X')
end
return Forest
```

Algorithm 2: $iTree(X)$

Complexity: Time - $O(\psi^2)$, Space - $O(\psi)$

Input: X - input data

Output: an $iTree$

```
1  q ← RandomChoice(X.attributes)
2  p ← RandomNumber(X[splitAttr].min(), X[splittAttr].max())
3  tree ← Node { left ← None, right ← None, size ← X.size
4                splitAttr ← q, splitVal ← p}
5  if X.size > 1 and X[splitAttr].numUnique() > 1 then
6  |  X_l ← X.where(q < p)
7  |  X_r ← X.where(q ≥ p)
8  |  tree.left ← iTree(X_l)
9  |  tree.right ← iTree(X_r)
10 end
11 return tree
```

2.3 Evaluation Stage

Algorithm 3: $PathLength(x)$

Complexity: Time - $O(t\psi)$, Space - $O(1)$

Input: x - input instance, T - an iTree, $hlim$ - height limit, e - current path length to be initialized to zero when called first time

Output: path length of x

```
1  if (T.right is None) and (T.left is none) and (e ≥ hlim) then
2  |  return e + c(T.size)                    // c(...) is defined in Equation 2.1
3  end
4  a ← T.splitAttr
5  if x[a] < T.splitVal then
6  |  return PathLength(x, T.left, hlim, e + 1)
7  end
8  else
9  |  return PathLength(x, T.right, hlim, e + 1)
10 end
```

In the evaluation stage[3], 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 $c(size)$ is defined in the next section.

2.4 Anomaly Score

The difficulty in deriving an anomaly 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.

Since iTrees have an equivalent structure to Binary Search Tree, the estimation of average $h(x)$ for external node terminations is the same as that of the unsuccessful searches in BST.

Section 10.3.3 of [3] gives the average path length of unsuccessful searches in BST as:

$$c(\psi) = \begin{cases} 2H(\psi-1) - 2(\psi-1)/n & \text{for } \psi > 2, \\ 1 & \text{for } \psi = 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

where $H(i) \approx \ln(i) + 0.5772156649$ (euler's constant), is the harmonic number. 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:

$$s(x, \psi) = 2^{\frac{-E[h(x)]}{c(\psi)}} \tag{2.2}$$

where $E[h(x)]$ is the average of $h(x)$ from a collection of iTrees.

Anomaly score $s(x, \psi)$ is interpreted as follows:

1. if $s \approx 1$, instance is abnormal
2. if $s \approx 0$, instance is nominal
3. if $s \approx 0.5$, no distinct anomaly

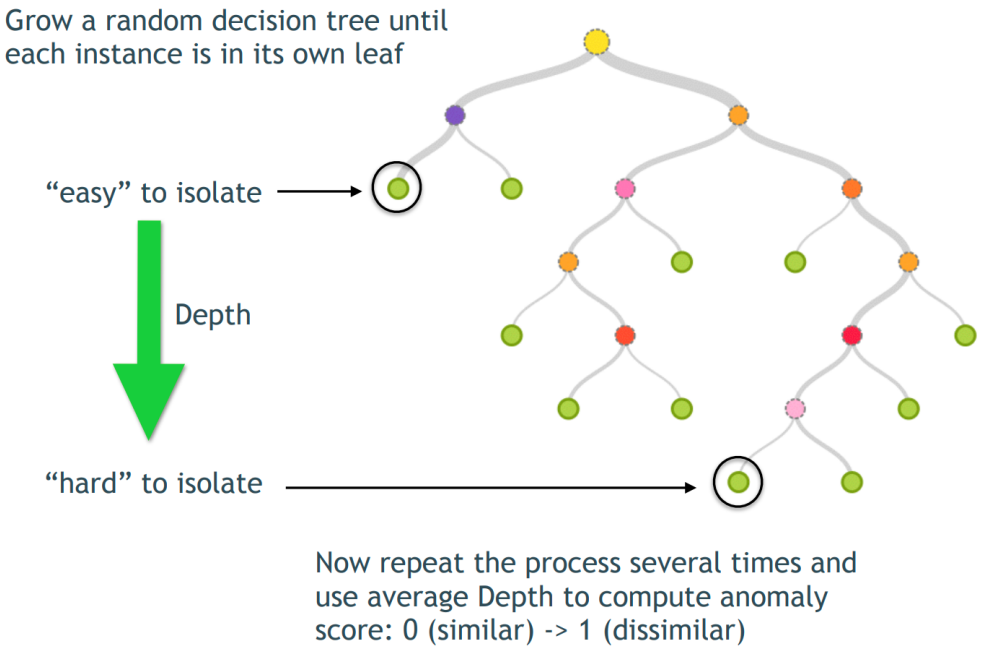


Fig. 2.3: Isolation Forest
Source: slideshare.com

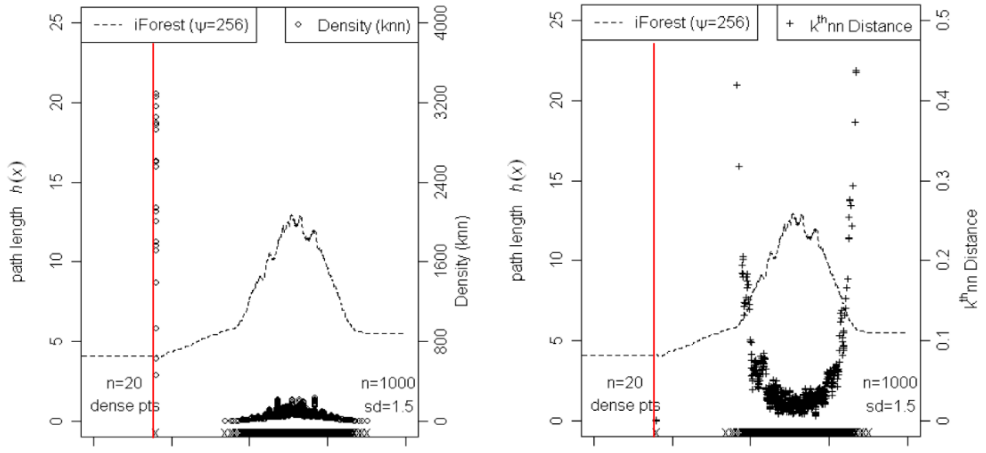
In practical use cases, one has to set a threshold deciding the result. And finding a good threshold is a very difficult task. Most of the times the anomalous points' score overlaps with the nominal points' score.

2.5 Isolation vs. Distance and Density

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

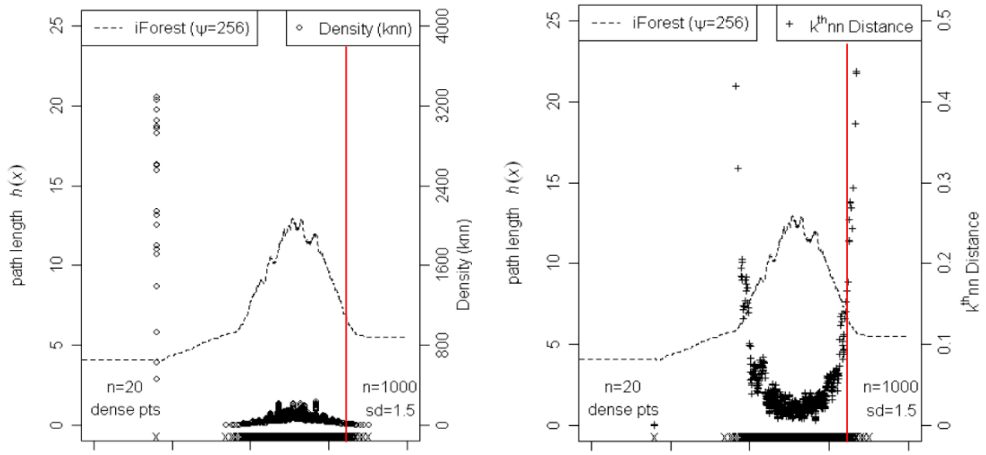
assumption is that 'Normal point is close to its neighbours and anomaly is far from its neighbours'.

There are violations of these assumptions: i) high density and short distance do not always imply normal instances ii) low density and long distance do not always imply anomalies.



(a) density based method k-NN (b) distance based method kth-distance

Fig. 2.4: High density and short distance do not always imply normal instances.
Source: Lui et al. [2]



(a) density based method k-NN (b) distance based method kth-distance

Fig. 2.5: Low density and long distance do not always imply anomalies.
Source: Lui et al. [2]

Isolation Forest compares favourably to distance and density based methods in terms

of accuracy and processing time. Distance and density based suffers immensly in terms of accuracy and processing time because of curse of dimensionalty.

Distance based methods also suffers from masking and swamping effect. In isolation forest, masking and swamping effects can be managed by adjusting the *hlim* parameter during evaluation. Refer section 4.5, 5.3 and 5.4 of Lui et al. [2].

Codes for Isolation Forest and Isolation Tree can be found at this repository.

2.6 Conclusion

To conclude, isolation forest is one of the most efficient and accurate methods for anomaly detection. It outperforms various density and distance based methods like ORCA, DOLPHIN, LOF, ROF, etc. and their variants in terms of accuracy and performance. Still there are some shortcomings of the isolation forest that we will discuss in next chapter.

Chapter 3

PIDForest

give details of your algorithm

3.1 Conclusion

In this chapter, we proposed a distributed algorithm for construction of xyz. The complexity of this algorithm is $O(n \log n)$. Next chapter presents another distributed algorithm which has linear time complexity based on xyz.

Chapter 4

Contributions

The algorithm presented in previous chapter has $O(n)$ time complexity. We further propose another distributed algorithm in this chapter based on xyz which has linear time complexity.

4.1 Construction

Write ...

4.2 Improved Method

Write...

4.3 Conclusion

In this chapter, we proposed another distributed algorithm for XYZ. This algorithm has both time complexity of $O(n)$ where n is the total number of nodes. In next chapter, we conclude and discuss some of the future aspects.

Chapter 5

AI in Image Compression

This chapter contains information about work done before COVID-19 pandemic lockdown.

5.1 Construction

Write ...

5.2 Improved Method

Write...

5.3 Conclusion

In this chapter, we proposed another distributed algorithm for XYZ. This algorithm has both time complexity of $O(n)$ where n is the total number of nodes. In next chapter, we conclude and discuss some of the future aspects.

Chapter 6

Conclusion and Future Work

write results of your thesis and future work.

References

[1] Pham H, *Handbook of Engineering Satistics*. Springer, 2006.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, Mar. 2012. [Online]. Available: https://doi.org/10.1145/2133360.2133363

[3] B. R. Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*. USA: John Wiley and Sons, Inc, 1998.