# Anomaly Detection via Isolation Forest Embedding

Manuel Salamino

*Advisor*: Prof. Giacomo Boracchi
*Co-advisor*: Filippo Leveni
*Academic year: 2020-21*

Politecnico di Milano

## Introduction

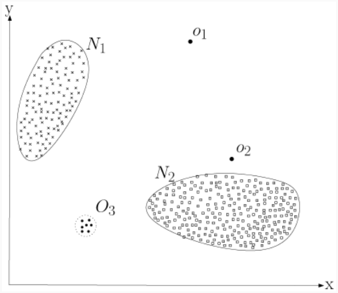
## Introduction to Anomaly Detection

**Anomaly Detection:** a Data Mining process with the aim of finding patterns in data that do not conform to expected behavior [1]

[1] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly detection: A survey*. 2009.

## Introduction to Anomaly Detection

**Anomaly Detection:** a Data Mining process with the aim of finding patterns in data that do not conform to expected behavior [1]



[1] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly detection: A survey.* 2009.

## Introduction to Anomaly Detection

**Anomaly detection** is used in many different fields, and *anomalies* usually represent *negative events*.

## Introduction to Anomaly Detection

**Anomaly detection** is used in many different fields, and *anomalies* usually represent *negative events*.

For example, anomalies in **credit card transaction** data could indicate credit card or identity theft.

## Introduction to Anomaly Detection

**Anomaly detection** is used in many different fields, and *anomalies* usually represent *negative events*.

For example, anomalies in **credit card transaction** data could indicate credit card or identity theft.

Or, in a **computer network** an anomalous traffic pattern could represent an hacked computer.

## Problem Formulation

The Anomaly Detection problem consists in monitoring a set of *n* data:

$$\mathcal{X} = \{x_1, \dots, x_n \quad | \quad x_i \in \mathbb{R}^d\}$$

each element $x_i$ is realization of a random variable having the following pdf

$$x_i \sim \begin{cases} \phi_0 & \textit{normal data} \\ \phi_1 & \textit{anomalies} \end{cases}$$

where $\phi_0 \neq \phi_1$ and $\phi_0$ and $\phi_1$ are **unknown**.

## Our Work

## Our Work

In this thesis we faced the problem by taking as starting point a milestone anomaly detection algorithm: **Isolation Forest**[2].

We use the **intermediate output** of Isolation Forest to create an **embedding**, hence a new data representation to be used in other anomaly detection techniques.

[2]F. T. Liu, K. M. Ting and Z. Zhou. *Isolation Forest.* 2008.

## Isolation Forest

## Isolation Forest

Isolation Forest (iForest) is a **breakthrough** unsupervised model-based method for anomaly detection.

## Isolation Forest

Isolation Forest (iForest) is a **breakthrough** unsupervised model-based method for anomaly detection.

Other methods:

$\rightarrow$ construct a *normal data profile* $\hat{\phi}_0$

- conform to $\hat{\phi}_0 \Rightarrow$ normal
- not conform $\Rightarrow$ anomaly

## Isolation Forest

Isolation Forest (iForest) is a **breakthrough** unsupervised model-based method for anomaly detection.

Other methods:

$\rightarrow$ construct a *normal data profile* $\hat{\phi}_0$

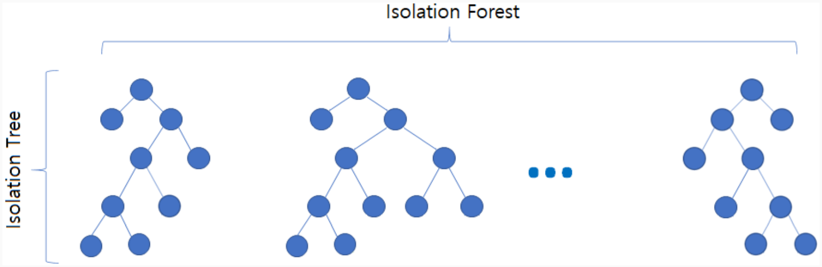
- conform to $\hat{\phi}_0 \Rightarrow$ normal
- not conform $\Rightarrow$ anomaly

iForest:

- anomalies are *few* and *different*
- directly isolates anomalies
- anomalies are susceptible to isolation

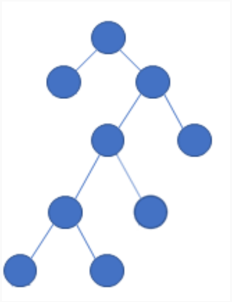
## Isolation Forest

The model is based on a trees ensemble, each tree is called *Isolation Tree* (iTree).



https://machinelearninggeek.com/outlier-detection-using-isolation-forests/

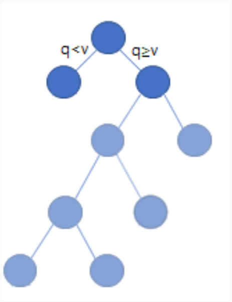
## Isolation Tree



Two nodes type:

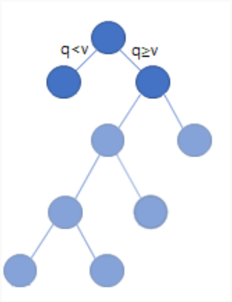
- **terminal** node, leaf with no child
- **internal** node, with exactly two child nodes (left and right)

## Isolation Tree



Each internal node has a *splitting criterion*

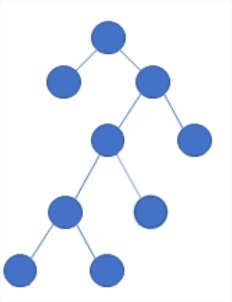
## Isolation Tree



Each internal node has a *splitting criterion*

- *splitting attribute* $q$ is randomly selected (among the $d$ possible)
- *split value* $v$ in randomly selected in range of values of $q$

## Isolation Tree



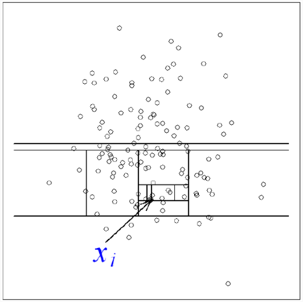
Repeat splits until:

1. the tree reaches a height limit $l$, or
2. $|X| = 1$, or
3. all data in X have the same value.

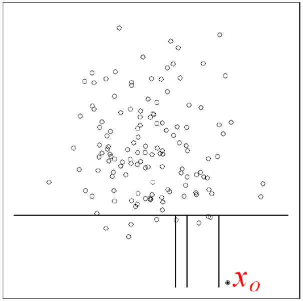
## Isolation Forest

iForest:

- anomalies are *few* and *different*
- directly isolates anomalies
- anomalies are susceptible to isolation



**(a)** Isolating normal.

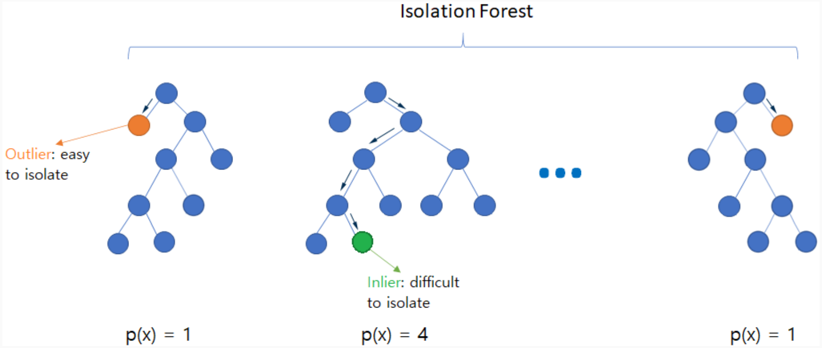


**(b)** Isolating anomaly.

F. T. Liu, K. M. Ting and Z. Zhou. *Isolation Forest*. 2008.

## Isolation Forest - Evaluation

Predicted label depends on the depth of the leaf on which *x* falls



https://machinelearninggeek.com/outlier-detection-using-isolation-forests/

## Anomaly Score

**Anomaly score** $s$ of an instance $x$:

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

where $E(p(x))$ is the average *path length* $p(x)$. $c(n)$ is a normalization factor and depends on samples $n$ used to train each iTree.

## Anomaly Score

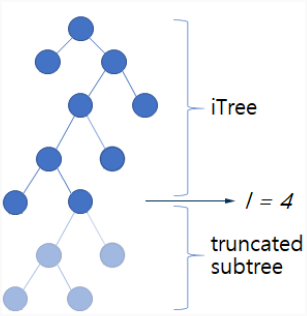
Anomaly score $s$ of an instance $x$:

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

where $E(p(x))$ is the average *path length* $p(x)$. $c(n)$ is a normalization factor and depends on samples $n$ used to train each iTree.

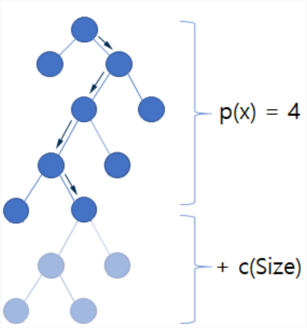
- $E(p(x)) \to 0 \quad \Rightarrow \quad s(x, n) \to 1 \quad \Rightarrow \quad$ anomalous instance;
- $E(p(x)) \to n - 1 \quad \Rightarrow \quad s(x, n) \to 0 \quad \Rightarrow \quad$ normal instance.

## Correction Factor



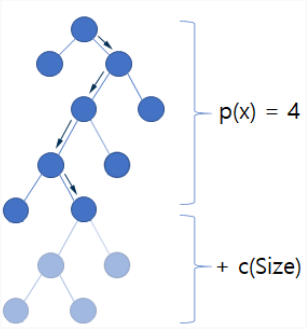
When tree height limit $l$ is reached, the iTree is truncated.

## Correction Factor



$c(Size)$ **estimates the truncated subtree height**, *Size* is the number of not split elements in that leaf node.

## Correction Factor



$c(Size)$ **estimates the truncated subtree height**, *Size* is the number of not split elements in that leaf node.

$c(n)$: estimate the height of a Binary Search Tree with $n$ elements

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

where $H(i) = ln(i) + 0.5772156649$.

## Proposed Solution

## Idea

Idea: use the intermediate output of iForest to create an embedding, hence a new data representation to be used in other anomaly detection techniques.

## Outline

Outline:

- Embedding definition
- New anomaly score
- Anomaly detection technique in embedding representation

## Embedding

## Embedding

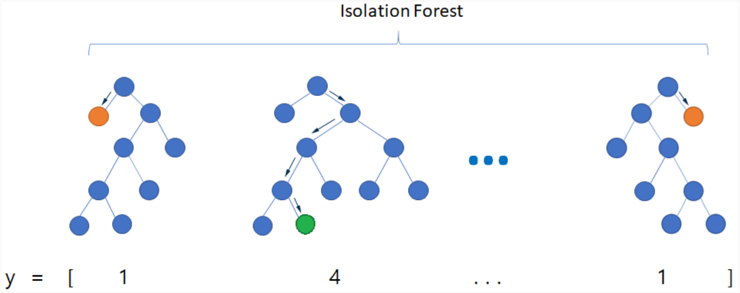
Some definitions for the embedding:

- **depths vector** $y$: intermediate output of iForest, $y \in \mathbb{R}^t$
  $t$: number of trees, $y_i$ is the returned depth of the *i-th* iTree

## Embedding

Some definitions for the embedding:

- **depths vector** $y$: intermediate output of iForest, $y \in \mathbb{R}^t$
  $t$: number of trees, $y_i$ is the returned depth of the *i-th* iTree



## Embedding

Some definitions for the embedding:

- **histogram** $h$: histogram of *depths vector y*,

$$h = \mathit{histogram}(y), \qquad \|h\|_1 = 1$$

$$h \in \mathbb{Q}^n, \qquad n = \mathit{ceil}(\mathit{max}(y))$$

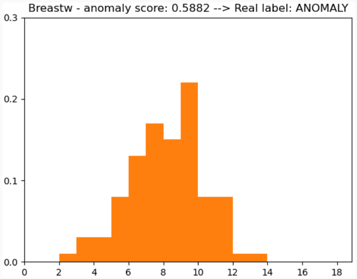
## Embedding

Some definitions for the embedding:

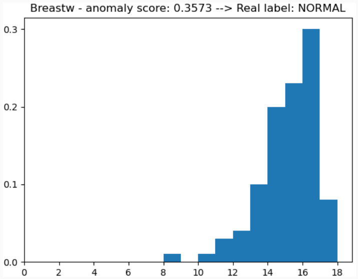
- **histogram** *h*: histogram of *depths vector y*,

$$h = histogram(y), \qquad \|h\|_1 = 1$$

$$h \in \mathbb{Q}^n, \qquad n=ceil(max(y))$$



**(a)** Anomalous instance.



**(b)** Normal instance.

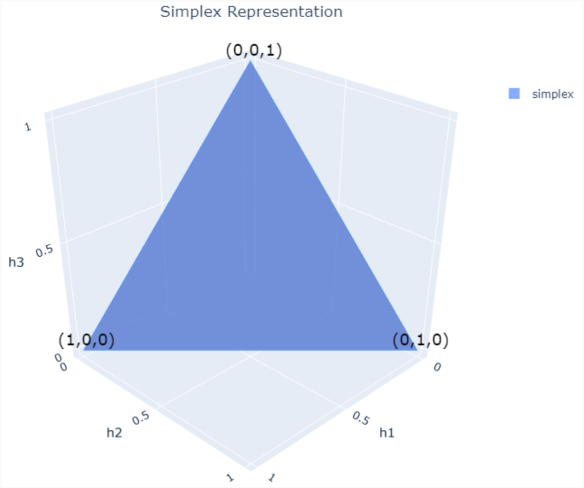
## Embedding

Let's summarize the steps:

$$x \in \mathbb{R}^d \xrightarrow{\textit{iForest}} y \in \mathbb{R}^t \xrightarrow{\textit{histogram}} h \in \mathbb{Q}^n$$

The **embedding** consists in the **histogram** representation of a set of data $X$.

## Embedding - Simplex



Embedding lies on **hyperplane** $h_1 + h_2 + ... + h_n = 1$.
$h_i \in [0, 1]$ constraints data points in this region of the hyperplane.

## Correction Factor in Embedding

The *correction factor* $c(n)$ leads to two problems in the embedding:

1. *increases the embedding dimension* (increases *n=ceil(max(y))* of $\mathbb{Q}^n$), and
2. *the depth becomes a real number*, $y \in \mathbb{R}^t$, instead of integer.

We **remove** the **correction factor** in order to avoid these two drawbacks.

## New Anomaly Score

## New Anomaly Score

**Idea:** find a new anomaly score starting from the embedding formulation of $E(p(x))$.

## New Anomaly Score - Average Path Length

In **traditional iForest** the average path length is computed as follows:

$$E(p(x)) = \frac{1}{t}\sum_{i=1}^{t} y_i$$

## New Anomaly Score - Average Path Length

In **traditional iForest** the average path length is computed as follows:

$$E(p(x)) = \frac{1}{t}\sum_{i=1}^{t} y_i$$

Using the **embedding** there is a new formulation:

$$E(p(x)) = \sum_{i=1}^{n} i \cdot h_i$$

This formulation can be seen as a linear combination.

## New Anomaly Score - Average Path Length

We can find new weights $\hat{w}_i$.

New weights $\hat{w}_i$ are computed using a supervised technique (*Linear Discriminant Analysis - LDA*) to test **how much margin for improvement** there is using linear combination.

## Embedding + LDA

## Linear Discriminant Analysis

The new formulation is:

$$L(y) = \sum_{i=1}^{n} \hat{w}_i \cdot h_i,$$

$\hat{w}_1, \ldots, \hat{w}_n$ weights computed using LDA.

We are computing an **upper bound** of the performance of iForest using linear combination of $h$, since the optimal weights $\hat{w}$ are *dataset-specific*.

## Embedding + LDA - Experiment

| *Dataset* | *iForest* | *Embedding+LDA* |
|---|---|---|
| Http | **1.000** | 0.999 |
| ForestCover | 0.938 | **0.969** |
| Mulcross | 0.899 | **0.957** |
| Smtp | 0.850 | **0.853** |
| Shuttle | 0.995 | **0.997** |
| Mammography | 0.764 | **0.823** |
| Annthyroid | 0.816 | **0.818** |
| Satellite | 0.707 | **0.726** |
| Pima | 0.631 | **0.638** |
| Breastw | 0.957 | **0.972** |
| Arrhythmia | **0.781** | 0.776 |
| Ionosphere | **0.863** | 0.856 |

*Embedding+LDA* shows that the **margin for improvement** is **small**.

## Anomaly Detection Technique in Embedding Representation

## One-Class SVM

We exploit the new framework in a different way.

We apply *One-Class Support Vector Machine* (OC SVM) in the embedding representation.

## Embedding + OC SVM - Experiment

| *Dataset* | *iForest* | *Embedding+OC SVM* |
|---|---|---|
| Http | **1.000** | 0.999 |
| ForestCover | **0.943** | 0.912 |
| Mulcross | 0.896 | **0.918** |
| Smtp | **0.878** | 0.872 |
| Shuttle | 0.995 | **0.997** |
| Mammography | 0.752 | **0.754** |
| Annthyroid | 0.823 | **0.828** |
| Satellite | 0.703 | **0.711** |
| Pima | 0.631 | **0.641** |
| Breastw | 0.957 | **0.966** |
| Arrhythmia | 0.780 | **0.787** |
| Ionosphere | **0.868** | 0.865 |

*Embedding+OC SVM* shows results that are **very similar** to **traditional iForest**, the differences are small (<0.01).

## Embedding + OC SVM - Experiment

Interesting insight on **OC SVM parameter** tuning.

Comparing *embedding+OC SVM* and *OC SVM on original data*, we note that *embedding+OC SVM* version has **much more stability** with different parameter values than *OC SVM on original data*.

## Embedding + OC SVM - Experiment

| *Dataset* | *kernel* | *gamma* | $\nu$ | *Embedding+OC SMV* | *OC SVM* |
|---|---|---|---|---|---|
| Pima | poly | auto | 0.9 | **0.618** | 0.243 |
| Pima | poly | auto | 0.1 | **0.618** | 0.307 |
| Pima | poly | scale | 0.9 | **0.618** | 0.243 |
| Pima | poly | scale | 0.1 | **0.618** | 0.307 |
| Pima | linear | auto | 0.9 | **0.618** | 0.237 |
| Pima | linear | auto | 0.1 | **0.618** | 0.292 |
| Pima | linear | scale | 0.9 | **0.618** | 0.237 |
| Pima | linear | scale | 0.1 | **0.618** | 0.292 |
| Pima | rbf | auto | 0.9 | 0.449 | 0.490 |
| Pima | rbf | auto | 0.1 | 0.547 | 0.530 |
| Pima | rbf | scale | 0.9 | 0.453 | 0.636 |
| Pima | rbf | scale | 0.1 | 0.552 | 0.669 |
| Pima | sigmoid | auto | 0.9 | **0.618** | 0.5 |
| Pima | sigmoid | auto | 0.1 | **0.618** | 0.5 |
| Pima | sigmoid | scale | 0.9 | **0.618** | 0.233 |
| Pima | sigmoid | scale | 0.1 | **0.618** | 0.274 |

## Conclusion

## Conclusion

We faced the problem of anomaly detection by using a state-of-the-art algorithm: iForest.

The main contributions are:

- use iForest as an **embedding** where we are able to represent data in a totally different way, based on the iForest intermediate output;
- **new algorithms** to perform anomaly detection, even if the margin for improvement is small.

## Future Works

Future works includes:

- Define the embedding using iForest *with correction factor*;
- Apply in the embedding space other anomaly detection techniques that can be applied also in original space.

Thank you for your attention!

## Backup Slides

## Embedding

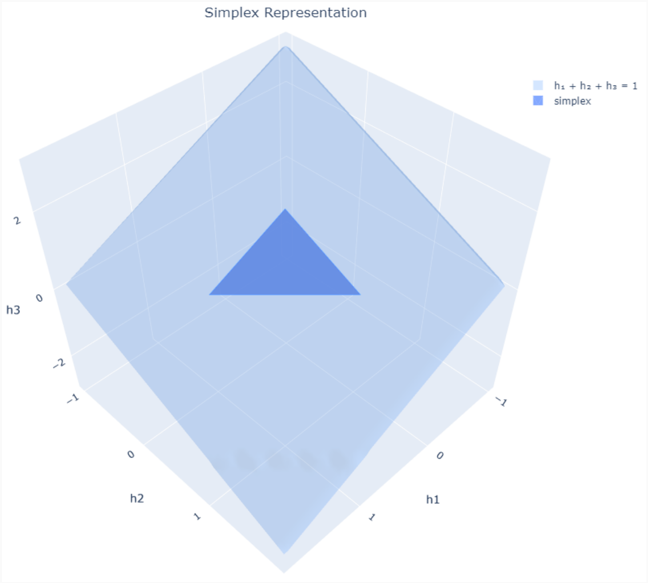
Since:

$$\|h\|_1 = 1 \qquad \Rightarrow \qquad \sum_{i=1}^{n} h_i = 1, \quad h_i \in [0, 1]$$

Considering $h_1 + h_2 + ... + h_n = 1$ we obtain an **hyperplane**.

And given the constraint $h_i \in [0, 1]$, for $i, \ldots, n$ the data points are limited in region of the hyperplane, a **simplex** (a n-dimensional triangle).

# Embedding - Simplex



## Remove Correction Factor - Experiment

**Goal.** *Check whether iForest with correction factor and iForest without correction factor have the same performance.*

We use *Paired-T-test* to check it.
The test is made on ROC AUCs obtained executing 10 times iForest *with correction* and *without correction*.

## Remove Correction Factor - Experiment

We perform two tests over all the dataset in [3]:

- *two-sided paired-T-test*, to check if the ROC AUCs of the two variants are **equivalent**. It shows a *p-value*=0.00313, and this means that the null hypothesis of identical averages is **rejected** and the two variants have different performances;
- *one-sided* to verify if **without correction performs better than with correction**, it returns a positive *t-statistic*, but a *p-value*=0.0007, hence **is rejected**.

We can state that **traditional iForest performs better than that without correction**. We have to take this into account, but to avoid the problems listed before, we use iForest *without correction*.

[3] F. T. Liu, K. M. Ting and Z. Zhou. *Isolation Forest.* 2008.

## Embedding + OC SVM - Experiment

For this experiment there is also an interesting insight on **OC SVM parameter** tuning. We test three parameters:

- **Kernel**, function to compute the similarity of two instances in feature space. Tested kernels: *linear*, *poly*, *rbf* and *sigmoid*.
- **Gamma**, how far the influence of a single training example reaches. Two predefined values: *scale* and *auto*.
- $\nu$, upper bound to the fraction of anomalies that can lie outside the boundary of normal region (e.g. if $\nu = 0.1$, at most 10% outside the decision boundary).

https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html