POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Anomaly Detection via Isolation Forest Embedding

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Manuel Salamino

Advisor: Prof. Giacomo Boracchi

Co-advisor: Filippo Leveni

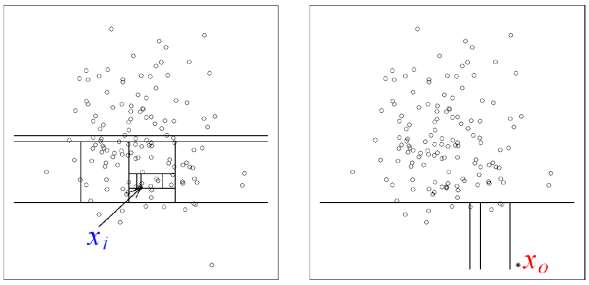
Academic year: 2020-2021

Introduction

Anomaly Detection (AD) is a Data Mining process and consists finding unusual patterns or rare observations in a set of data. Usually anomalies represent negative events, in fact anomaly detection is used in many different fields, from medicine to industry. We faced the problem by taking as starting point a milestone AD algorithm: Isolation Forest[1] (iForest). This thesis propose to use the intermediate output of iForest to create an embedding, hence a new data representation on which known classification or anomaly detection techniques can be applied. Our empirical evaluation shows that our approach performs just as well, and sometimes better, than iForest on the same data. But our most important result is the creation of a new framework to enable other techniques to improve the anomaly detection performance.

1. Isolation Forest

iForest[1] is an unsupervised model-based method for anomaly detection. This method represent a breakthrough, before iForest the usual approach to AD problems was: construct a *normal data profile*, then test unseen data instances and identify as anomalies the instances that do not conform to the normal profile. iForest differs from all the previous ones since it is based on the idea of directly isolates anomalies, instead of recognized them as far from the normal data profile.



(a) Isolating normal. (b) Isolating anomaly.

Figure 1: (a) Normal point x_i requires twelve partitions to be isolated; (b) Anomaly x_0 requires only four partitions to be isolated[1].

This approach works because anomalies are more susceptible to isolation than normal instances. In fact, Figure 1 shows that a normal instance requires much more partitions than an anomaly to be isolated. iForest assigns an anomaly score to each instance based on the number of splits required to isolate them.

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

is built on a different sub-sample of the dataset. This allows to obtain a set of different *experts*, each one able to recognized different type of anomalies (depending on the sub-sample on which it is trained). Then, the prediction of a test instance is made by computing the average of the paths covered in each iTree before reaching a leaf node. Shortest paths (few splits) identify anomalies, while longest ones (more splits) predict normal instances.

1.1. Isolation Tree

An iTree has the same structure of a Proper Binary Tree, a tree in which each node belongs exactly to one of the following categories:

- *external node*, a leaf node, no child;
- *internal node*, a node with exactly two children nodes (right and left).

Each iTree is built starting from a subset X of the dataset (because of sub-sampling):

$$|X| = \psi \qquad X \subset \mathbb{R}^d,$$

with ψ sub-sample size and d number of features. To build an iTree, as for a Binary Tree, we recursively divide X using as *splitting criterion* to divide data into right and left child. The *splitting criterion* consists in randomly selecting an attribute q (among the d possible) and a split value v in order to test if $q < v$. All the instances for which the test returns True are put in the left child, whereas all the others in the right one. The *split value* of v is randomly selected in the range of values of q in X.
The splits are repeated until one of the following conditions is reached:

(i) the tree reaches a height limit l, or

(ii) $|X| = 1$, or

(iii) all data in X have the same value.

1.2. Anomaly Score

To understand the anomaly score computation, we have to introduce the *Path Length* $p(x)$ of an instance x, which is defined as the number of edges traversed by x from the root node to the external node reached following the iTree rules defined in the training phase. The expected behaviour of normal instances is to exit the iTrees from the deepest leaves (longer path length), while anomalies are expected to exit from the shallowest ones (shorter path length).
Before defining the *anomaly score*, we define a function that is used both in anomaly score and in the evaluation phase: $c(n)$, which estimate the height of a Binary Search Tree[1] with n elements,

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

where $H(i)$ is the harmonic number and it can be estimated by $ln(i) + 0.5772156649$ (Euler's constant). Now we define the *anomaly score* s of an instance x as:

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

where $E(p(x))$ is the average $p(x)$ of the iTrees of the Forest and $c(\psi)$ is used to normalize.
The anomaly score reflects the relation between $E(p(x))$ and predicted label defined before: if shorter path length, s close to 1, the test instance is an anomaly, if longer path length, s close to 0, the test instance is normal.

1.3. c(n) as Correction Factor

During *evaluation* of an instance x, in path length computation, there is key passage on which we discuss later: when a node, during training, is defined as *terminal* because the tree height limit l is reached, the iTree is truncated, so, during evaluation, we add to $p(x)$ an adjustment $c(n)$, where n is the number of not split elements in that leaf node. $c(n)$ estimates the height of the truncated subtree using (1).

2. Proposed Solution

2.1. Problem Formulation

Let's look at the problem of AD in a more formal way[2]. The anomaly detection problem consists in monitoring a set of n data:

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

where each element x_i is realization of a random variable having pdf ϕ_0, with the aim of detect outliers, i.e. points that do not conform with ϕ_0.

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

where $\phi_0 \neq \phi_1$ and ϕ_0 and ϕ_1 are unknown. Since X has no labels we are in an unsupervised setting, furthermore there is no clear distinction between training and test data. Each dataset contains both normal and anomalous samples. The only mild assumption we can make is that normal data far outnumber anomalies.

2.2. Embedding

We introduce a new embedding that gives to input data a new representation, but first of all introduce some definitions:

- **depths vector** y: intermediate output of iForest, $y \in \mathbb{R}^t$. y_i is the returned depth of the *i-th* iTree;
- **histogram** h: histogram of *depths vector* y. Then it is normalized: $\|h\|_1 = 1$, $h \in \mathbb{Q}^n$, with n=*ceil(max(y))*, thus the first integer bigger than the *max* of y.
  More precisely each $h_i \in [0, 1]$ because represents the fraction of the t elements that are into a specific bin:

$$h = histogram(y), \qquad \|h\|_1 = 1 \tag{3}$$

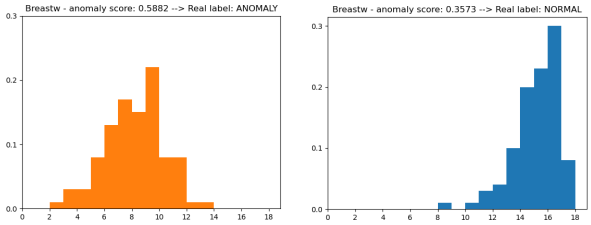
Let's summarize how to obtain the histogram h from input instance x:

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

Our *embedding* is a new dimensional space in $\mathbb{Q}^n$, where input instances x, transformed in the corresponding histogram h, lie in a n-dimensional *simplex* (a triangle in n dimensions), because the constraint $\|h\|_1 = 1$ positions all of them in an hyperplane, while $h_i \in [0, 1]$, for $i, \ldots, n$, constraints them in a simplex on that hyperplane.
Using the embedding, and so using this new representation of input data, we expect normal and anomalous instances to yield different histograms, i.e. anomalous instances have high frequencies for bins representing low depths, while normal instances have high frequencies for bins representing high depths. See the histograms in Figure 2.

2.3. Remove the Correction Factor

In Section 1.3 we introduce that, when an input instance x reaches a terminal node, to $p(x)$ is added the correction factor $c(Size)$, which estimates the height of the truncated subtree.



(a) Anomaly instance (b) Normal instance

Figure 2: Data coming from Breastw dataset. (a) Anomaly shows that most iTrees return a depth between 5 and 12. (b) Normal shows highest frequencies between 13 and 18.

The correction factor leads to two problems in the embedding:

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

We want to start the improvements by redefining the anomaly score (2) in the embedding, to do that is necessary to have $y \in \mathbb{N}^t$, so from this point on, the embedding is defined using depths vector $y \in \mathbb{N}^t$ computed using iForest *without correction*. Later, a formulation of $E(p(x))$ in the embedding if there is no correction factor.

2.4. Average Path Length in Embedding

We redefine the average path length, so the $E(p(x))$ of iForest, in the embedding, is computed with the following expression:

$$E(p(x)) = \sum_{i=1}^{n} w_i \cdot h_i, \tag{4}$$

where $w_i = i$, h_i is the *i-th* element of h and $h = histogram(y)$, as defined in (3).
Since we are considering iForest *without correction*, the depths vector y is composed only by integer values ($y \in \mathbb{N}^t$), thus h_i is the fraction of elements of the depths vector y that are equal to i. Hence by multipling the fraction of how many times a value is present in y (h_i) times the value itself (w_i) for $w_i = 1, \ldots, n$ we obtain the average path length.

2.5. Linear Discriminant Analysis

Looking at (4), we note that the $E(p(x))$ in the embedding is simply a linear combination of the

embedding feature $h_1, \ldots, h_n$ with predefined weights. The intuition is about finding a new weight vector $\hat{w}$, which is dataset-specific and can increase the AD performance. To this purpose we use Linear Discriminant Analysis (LDA) to find the new weights. LDA is a supervised method used in statistics and pattern recognition to find a linear combination of features that separates two or more classes. Hence we use LDA to find a linear combination of $h_1, \ldots, h_n$ which better separates the two classes.
The new formulation is:

$$L(y) = \sum_{i=1}^{n} \hat{w}_i \cdot h_i, \tag{5}$$

$\hat{w_1}, \ldots, \hat{w_n}$ weights computed using LDA.
Using LDA we are using a supervised technique, so we are computing something like an upper bound of the performance of iForest, since the linear combination is computed using optimal weights $\hat{w}$.

2.6. One-Class SVM

Another idea comes from the fact that the embedding is a totally new representation of the data. We apply in this new framework a well known AD technique: One-Class Support Vector Machine (OC SVM). It is able to identify the best boundary, in feature space, which separates normal instances from anomalies. We want to know if in this new data representation makes easier perform anomaly detection.

3. Experiments

3.1. Datasets

The datasets used for testing are the same used to evaluate iForest in [1]. Twelve datasets: eleven of them are natural datasets, plus a synthetic one. These are the same datasets used to test iForest, since iForest is the benchmark for our performance and we want to evaluate our modification proposals using the same data on which it was originally tested.

3.2. Remove the Correction Factor

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

Paired-T-test is used to determine whether the mean difference between two sets of observations is zero. The two sets of observation are the ROC AUCs obtained executing 10 times iForest *with correction* and *without correction*.
We use this to check if the performance are identical or not and if there is a statistical relevance on that. We perform two tests:

- *two-sided paired-T-test*, to check if the mean difference of the two variants is zero across different runs and datasets. 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 perfomances;
- *one-sided* to verify if *with correction* < *without correction*, it returns a positive *t-statistic*, but a *p-value*=0.0015, that halved become *p-value*=0.0007, hence **is rejected**.

We can state that traditional iForest performs better than the one *without correction*. We have to take this into account, but to get en embedding with low dimensional space and a depths vector y with integer values, we populate the embedding with iForest outputs generated using *without correction* version.

3.3. Embedding + LDA

Goal. Use Linear Discriminant Analysis (LDA) to find the best linear combination of embedding features that separates the two classes.

The idea of use LDA to find the weigths $\hat{w_1}, \ldots, \hat{w_n}$ comes from the intuition described in Section 2.5. The resulting optimal $\hat{w}$ is used in Formula (5), in order to obtain the scores and compute the ROC AUC. For every dataset, iForest is executed 10 times, and after each run, are computed: a) anomaly scores using average path length (standard iForest, no embedding) and b) anomaly scores computed using linear combination defined in (5). LDA performance are measured by 5-fold cross validation. Results are reported in Table 1.
Among the 12 tested datasets, in 9 of them the version Embedding+LDA has performance better than the standard iForest. The other good thing is that in the three datasets in which standard iForest performs better, the difference is very small, while the proposed version, in some

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

Table 1: In bold the higher ROC AUC. The embedding+LDA variant shows better results in 9 out of 12 datasets.

Dataset	iForest	Emb.+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

Table 2: In bold the higher ROC AUC. The embedding+OC SVM variant shows better results in 8 out of 12 datasets.

datasets (i.e. Mulcross, Mammography), has better results with a significant gap.
An important fact to remark is that the optimal weights $\hat{w}$ are *dataset-specific*: we couldn't find a vector $\hat{w}$ that reaches very good performance in all the dataset. So, for each dataset is necessary a LDA training phase to find the best weights for that domain. The fact that some datasets show results that are better in traditional iForest than in the embedding+LDA, is surprising because we expect that use a supervised techniques, trained on each single dataset, returns better performances.

3.4. Embedding + One-Class SVM

Goal. Obtain improvements applying One-Class SVM in the embedding.

With this experiment we evaluate the performance of OC SVM applied in the embedding space. The following results (Table 2) are obtained by executing 10 runs of iForest on each dataset, and after each run, are computed: a) the standard iForest performance and b) the embedding+OC SVM performance. In b), once the embedding representation is computed, OC SVM performances are measured using 5-fold Cross Validation.
The results show that in 8 out of 12 datasets the version Embedding+OC SVM has performance better than standard iForest, even if these differences are small (<0.01), except for *Mulcross* for which the difference, in favour of Embedding+OC SVM, is 0.022, and *ForestCover* for which the performance of standard iForest are better for 0.031. For this experiment there is also an interesting insight on OC SVM parameter tuning. We tuned three parameters defined in the scikit-learn documentation[3]:

- *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*.
- ν, 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).

Comparing the results of *embedding+OC SVM* and *OC SVM on original data* obtained by using different paramters combinations, we note that *embedding+OC SVM* version has much more stability with different parameter values than *OC SVM on original data*. We execute OC SVM by changing the values of the three parameters

and we obtain this behaviour in all the datasets. For the *embedding+OC SVM* the obtained ROC AUC differs from a factor of 10^{-4} or 10^{-5} from one parameters combination to another (except for *rbf kernel* which returns performance that are quite worst in all the datasets), while looking at *OC SVM in original data*, these differences are much bigger and the ROC AUCs change a lot using different *kernels* (10^{-2} or 10^{-3}).
Finally, we can infer that the embedding project input data in such a way that OC SVM perform in a similar way even if it is executed with different parameters, and this behaviour highlight a higher stability.

4. Conclusions

In this thesis we faced the problem of AD by using a state-of-the-art algorithm: iForest. In particular, the main goal was to change its last operations with the intention of replace them with more complex ones, able to capture new features in the data.
The main contributions are:

- using iForest as an embedding (Section 2.2) where we are able to represent data in a totally different way, based on the iForest output;
- new algorithms (Sections 2.5 and 2.6) to perform anomaly detection that reach results better than the starting point iForest.

Future works includes: embedding defined using iForest *with correction factor*. A future development can be an in-depth analysis of how to maintain the correction factor when the output of Isolation Forest is obtained and the embedding space is populated by input transformed data. The computation of the maximum value of the correction factor can be useful to determine the upper bound of the embedding dimension, and work with histograms with an higher number of bins probably give more information, even if the increasing dimensionality is a drawback. Another one is change the embedding formulation. The idea of starting from iForest output to create a new representation is maintained, but the histogram can be computed by using different bins, or giving more importance to the output of specific iTrees, instead of using an aggregative measure as the histograms. The last one is, given the embedding space, use it to apply other AD techniques that can be applied also in original space.

References

[1] Zhi-Hua Zhou, Fei Tony Liu, and Kai Ming Ting. *Isolation Forest*. 2008.

[2] Giacomo Boracchi and Diego Carrera. *Anomaly Detection in Images*. pages 17–22. ICIAP, September 2019.

[3] Scikit-Learn. *OneClassSVM Documentation*. URL `https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html`.