Proximity Isolation Forests

Antonella Mensi[1] Manuele Bicego[1] David M.J. Tax[2]

[1]University of Verona, Italy

[2]TU Delft, The Netherlands

January 12, 2021

Motivation

- Isolation Forests: successful Random-Forest based methodology for outlier detection (Liu, 2008).
- Isolation Forests + extensions → vectorial data only.
- Many outlier detection tasks that work with non-vectorial data:
  - Sequences, e.g. finding abnormal sequences in an ECG (Lourenço, 2013)
  - Images, e.g. detecting small masses in a brain image (El, 2016)
  - Graphs e.g. detecting traffic-related anomalies (Shekhar, 2001)

↓

There is no RF-based methodology to solve these tasks!

Proposal

There exist several measures for measuring the distance between sequences, images, etc.

We can employ pairwise distances to work directly with non-vectorial data.

Our proposal: Proximity Isolation Forests (PIF)  RF-based methodology for outlier detection. It works with trees that are built using only *pairwise distances* between the objects.

Proximity Isolation Tree

- Proximity Isolation Tree (PIT): built recursively using a matrix D containing all pairwise distances $d(\cdot, \cdot)$ between the objects in dataset $\mathcal{S}$.

- (*Testing*) How does an object x traverse a PIT? Two ways.
  1. For each internal node n we have one prototype P and a threshold θ. If $d(x, P) \leq \theta$ then $x \longrightarrow n_L$ otherwise $x \longrightarrow n_R$.
  2. For each internal node n we have two prototypes P_L and P_R. If $d(x, P_L) \leq d(x, P_R)$ then $x \longrightarrow n_L$ otherwise $x \longrightarrow n_R$.

where L and R stands for left and right.

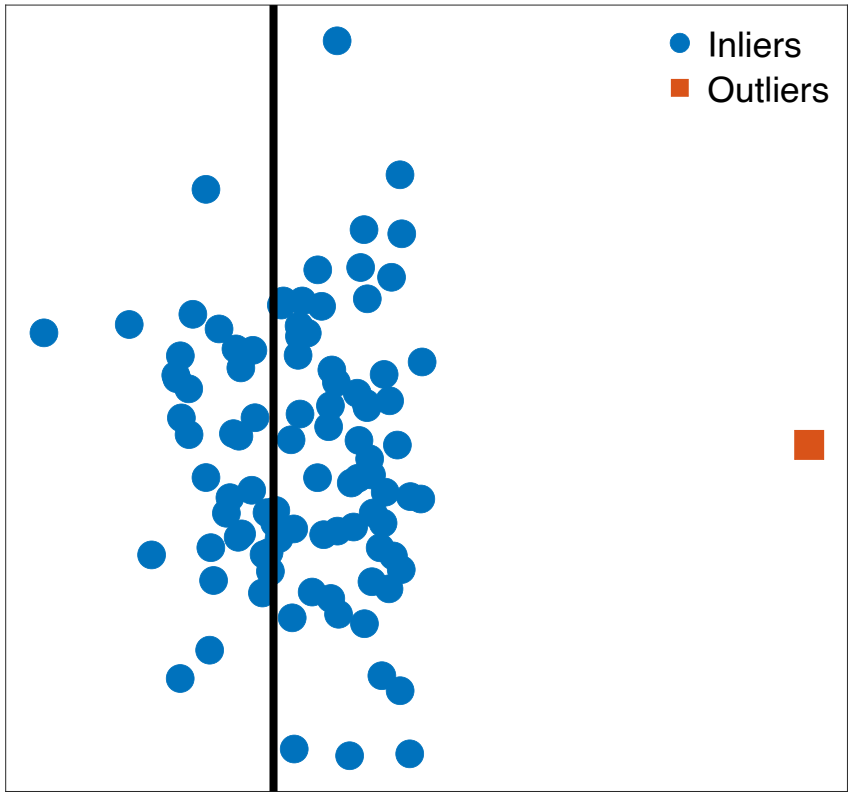
Proximity Isolation Tree: Splitting Criteria

(Training) When building a PIT, we have to find the best split in each internal node ➡ Five ways: *two* random and *three* optimized.

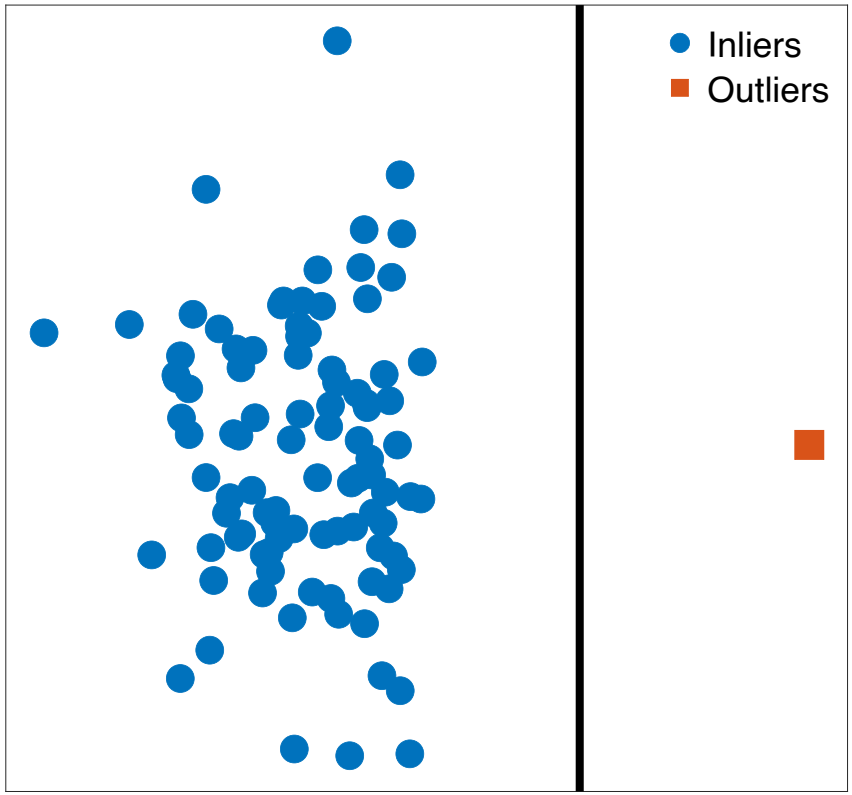
1. **R-1P**: Select *randomly* one prototype P among the objects in n. Then pick randomly a threshold θ in the range $[\min_{x \in n} d(x, P), \max_{x \in n} d(x, P)]$.
2. **R-2P**: Choose *randomly* a pair of objects as prototypes P_L and P_R among the objects in n.

Proximity Isolation Tree: Splitting Criteria

- Isolation of outliers ➡ big reduction in variance.



Smaller decrease in variance



Higher decrease in variance

Proximity Isolation Tree: Splitting Criteria

- No features ⇒ no variance.
- Necessary to define a measure that captures the sparseness of the distance values ⇒ scatter.
  1. *ScatterD*

$$S_D(\mathrm{D}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(i,j)$$

  2. *ScatterP*

$$S_P(\mathrm{D}, P) = \frac{1}{N} \sum_{i=1}^{N} d(i,P)$$

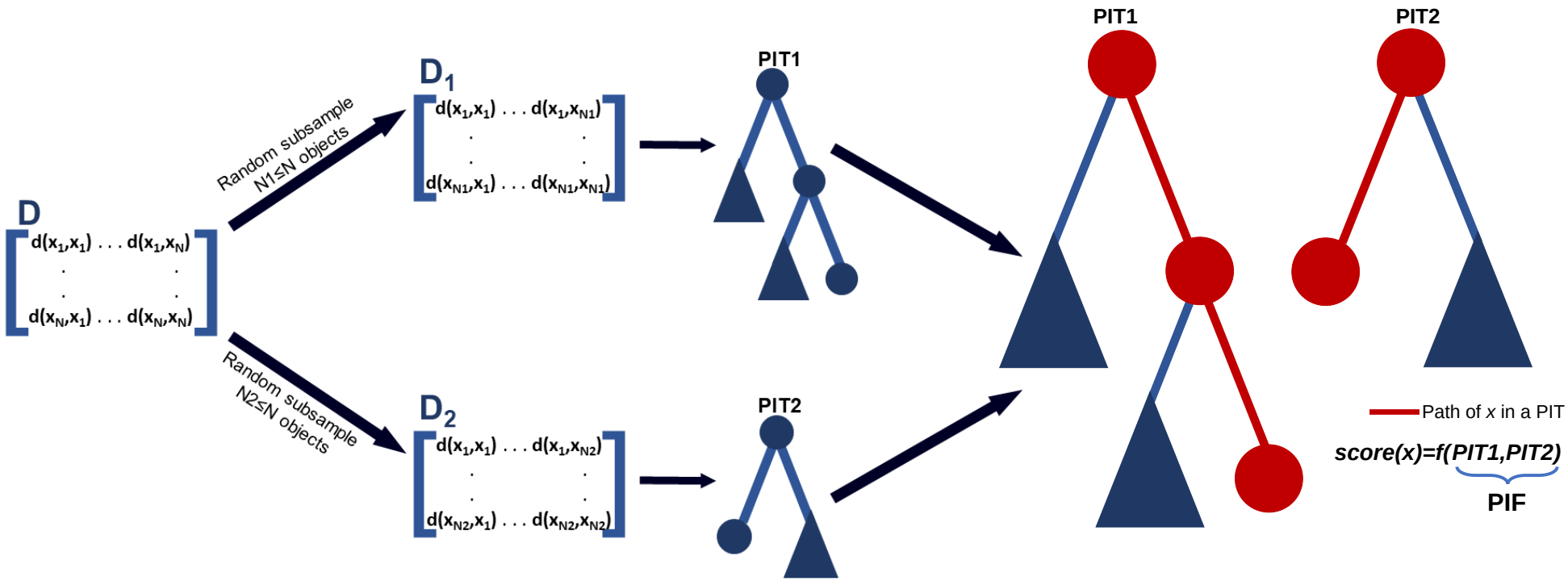
where N is the number of objects in D and P is a prototype.

Proximity Isolation Tree: Splitting Criteria

Optimized criteria:

3. **O-1PS$_D$** Choose the pair (P, θ) which optimizes *ScatterD* where P is an object in n and $\theta \in \{d(x, P) | x \in n\}$.
4. **O-2PS$_D$**: Choose the pair (P_L, P_R) which optimizes *ScatterD* where P_L and P_R are objects in n.
5. **O-2PS$_P$**: Choose the pair (P_L, P_R) which optimizes *ScatterP* where P_L and P_R are objects in n.

Proximity Isolation Forests



Datasets[1]

Dataset	Nr. Obj.	% of Outl.	Dist. Type
DelftPedestrians	689	3.92%	Cloud Dist.
DelftGestures	1500	5%	Dynamic Time Warping
WoodyPlants	791	7.96%	Shape Dist.
Pendigits	10992	9.60%	Weighted Edit Dist.
Zongker	2000	10%	Deformable Template Dist.
ChickenPieces (V1)	446	13.68%	Weighted Edit 2D Shape Dist.
Protein	213	14.08%	Evolutionary Dist.
Flowcyto (V1)	612	54.74%	Histogram Dist.

[1] Source: `http://prtools.tudelft.nl/Guide/37Pages/distools.html`

Experimental Setup

- Number of split variants: *5 options*, **R-1P**, **R-2P**, **O-1PS**$_D$, **O-2PS**$_D$, and **O-2PS**$_P$.
- Number of trees in a forest T: 50, 100, 200, 500.
- Number of training samples used to build a forest N: 64, 128, 256, 512.
- Maximum depth per tree D: $N-1$, $D = \log_2(N)$.
- Each experiment repeated 10 times.
- AUC as accuracy measure.

Experimental Observations

- There is a best option for each parameter.
- Each dataset has a preferred variant. The preferred variant seems to vary based on the outlier percentage of the dataset. From lowest to highest: **R-1P**, **O-2PS**$_P$, **O-2PS**$_D$ and **R-2P**.

Complete and detailed results can be found in the paper.

Comparison with classical density and distance-based methodologies

Dataset	NNd	KNNd	LOF	LOF-Range	K-Cent.	PIF
DelftPedestrians	0.524	0.567	0.553	0.579	0.629	**0.799** (*0.799*)
DelftGestures	0.419	0.440	0.547	0.579	0.643	**0.955** (*0.976*)
WoodyPlants	0.451	0.390	0.659	0.639	0.714	**0.910** (*0.930*)
Pendigits	0.505	0.490	0.492	0.466	0.600	**0.745** (*0.755*)
Zongker	0.566	0.476	0.564	0.514	0.752	**0.796** (*0.811*)
ChickenPieces	0.462	0.462	0.456	0.444	NaN	**0.825** (*0.846*)
Protein	0.413	0.820	0.922	0.919	0.861	**0.984** (*0.985*)
Flowcyto	0.498	0.448	0.619	0.623	0.629	**0.708** (*0.737*)
Average	0.479	0.524	0.602	0.596	0.688	**0.840** (*0.855*)

- Chosen PIF parametrization: $T = 200$, $D = log_2(N)$, $N = 256$, Variant=$O - 2PS_D$.[2]

[2] In italics we report the *best* result per dataset.

Conclusions

- PIF: RF-based methodology for outlier detection that works with non-vectorial data.
- PIF only needs a pairwise distance matrix as input.
- Five different criteria (random and optimized) to choose the best split in a node.
- Good results on 8 datasets, also when compared to classical techniques.

Thank you for your attention!