NDD: Outlier Detection

Created	@Oct 22, 2019 11:32 PM
Reviewed	☐
Semester	Fall19
Syllbus	https://github.com/NeuroDataDesign
Type	Lecture

NDD Links: Google drive, Resource, nbviewer, GithubID

Note: Oct 21, 2019

▼ Goals

- Benchmark Isolation Forest (IF) and Extended Isolation Forest (EIF)
- Apply Algo2-3 from EIF paper to USPORF

▼ Citations

1. Web Iris dataset

2. Paper Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413-422). IEEE.
3. Paper Hariri, S., Kind, M. C., & Brunner, R. J. (2018). Extended Isolation Forest. *_arXiv preprint arXiv_*:1811.02141.
4. Web sckit-learn, Isolation Forest
5. Github, Extended Isolation Forest
6. Web Outlier Detection DataSets (ODDS): Vertebral dataset, Wine dataset

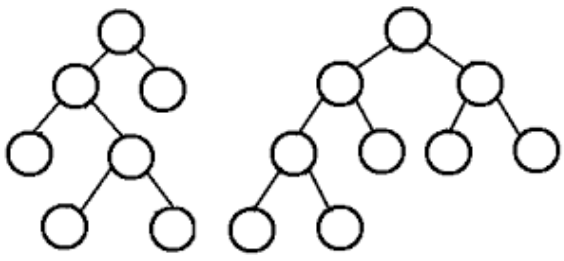
▼ Paper: Isolation Forest

▼ **1. Definition: Isolation Tree (iTree)**

- *T* is divided into 2 daughters nodes *T_l,T_r*
- stop when reach (i) height limit, (ii) $|X| = 1$ in external nodes (iii) data in *X* have same value

Fig 1: iTree is proper binary tree= each internal node has 2 daughters

- external (ending) nodes= $n = |X|$
- internal (starting and latent) nodes = $n-1$
- tot nodes = $2n-1$
- Thus Tree grow linearly



▼ **2. Definition: Pathlength** *h(x)*

- *h(x)* = number of edge from root node to terminal node ~ avg. depth unsuccessful search in binary tree?

- *max h(x) ~ n, E[h(x)]=log n*

▼ Paper: Extended Isolation Forest

▼ **Algo 1**:

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

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

- *l* = average tree height. We are only interested length <*l*
- \psi ~ 256 is enough, \t ~ 100 is enough
- *Forest* = {*iTree*}, *i = 0,..,t* → see **Algo 2**
- *X'* (subset of *X with sample size \psi*)

$$\text{complxity} \sim O(t\psi \log \psi)$$

▼ **Algo 2**: Splitting algorithm

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

Input: X - input data, e - current tree height, l - height limit
Output: an iTree

1: **if** $e \geq l$ or $|X| \leq 1$ **then**
2: **return** $exNode\{Size \leftarrow |X|\}$
3: **else**
4: randomly select a normal vector $n \in \mathbb{R}^{|X|}$ by drawing each coordinate of $\vec{n}$ from a uniform distribution.
5: randomly select an intercept point $p \in \mathbb{R}^{|X|}$ in the range of X
6: set coordinates of n to zero according to extension level
7: $X_l \leftarrow filter(X, (X - p) \cdot n \leq 0)$
8: $X_r \leftarrow filter(X, (X - p) \cdot n > 0)$
9: **return** inNode$\{ Left \leftarrow iTree(X_l, e + 1, l),$
$Right \leftarrow iTree(X_r, e + 1, l),$
$Normal \leftarrow n,$
$Intercept \leftarrow p\}$
10: **end if**

- 6: extension level =0 → Standard IF,
- 6: extension level =1 → Extended IF,

▼ Splitting Algorithm

1. hyperplane slope (gradient)

$$\hat{n} \text{ from } \mathbb{R}^{|X|} \text{ sphere}$$

2. random intercept for the cut (chosen from the training data)

$$\vec{p} \text{ from } \{\vec{x} | \vec{x} \in X, \text{current node member}\}$$

1. and 2. create the splitting criteria

$$(\vec{x} - \vec{p}) \cdot \hat{n} \leq 0, \;\; \vec{x} \in X, \text{current node member}$$

- ≤ go left daughter node

- > go right daughter node

▼ Algo 3:

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

Input: x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x

1: **if** T is an external node **then**
2: **return** $e+c(T.size)$ {$c(.)$ is defined in Equation (2)}
3: **end if**
4: $n \leftarrow T.Normal$
5: $p \leftarrow T.Intercept$
6: **if** $(x-p) \cdot n \leq 0$ **then**
7: **return** $PathLength(x, T.left, e+1)$
8: **else if** $(x-p) \cdot n > 0$ **then**
9: **return** $PathLength(x, T.rigth, e+1)$
10: **end if**

- *T = iTree(X',e,l)* from **Algo 2**
  - T = { *exNode*{ pop $|X|$ in the node}, *lnNode*{*Normal*, *Intercept*, *left*, *right* } }
- eqn (2): *H(i),* harmonic number = ln*(i) +0.5772156649 ?*

$$c(n) = 2H(n-1) - (2(n-1)/n)$$