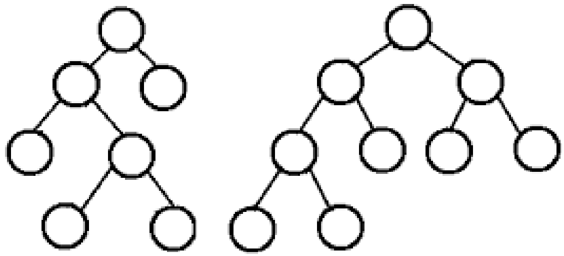
Note: Building trees and splitting algorithms

Note: Oct 28, 2019

- Reference:
    1. 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.
    2. Paper Hariri, S., Kind, M. C., & Brunner, R. J. (2018). Extended Isolation Forest. _*arXiv preprint arXiv*_:1811.02141.
    3. Paper: MORF, USPORF
- Goals
    - Apply Algo2-3 from EIF paper to USPORF
- Paper1: 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= $n = |X|$
        - dimension = N
        - internal (starting and latent) nodes = $n-1$
        - tot nodes = $2n-1$
        - Thus Tree grow linearly



full tree complete tree

▼ **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*

▼ Paper2: 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*

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

- *l* = average tree height. We are only interested length <*l*
- \psi ~ 256 is enough, \t ~ 100 is enough
- *Forest = {iTree},* *i = 1,..,t-1* → 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 ={0,1} → {Standard IF, Extended IF}

▼ Splitting Algorithm

1. normal vector (gradient), choose from uniform distribution in **R^N**

$$\hat{n} \in \mathbb{R}^N, N\text{-dimension}$$

2. random intercept for the cut choose from uniform distribution in **X**

$$\vec{p} \in \mathcal{X}, \mathcal{X} \text{ current node member}\}$$

1. and 2. create the splitting criteria

$$(\vec{x} - \vec{p}) \cdot \hat{n} \leq 0, \quad \vec{x} \in \mathcal{X}$$

- ≤ go left daughter node
- > go right daughter node

▼ Return

- **While** {if *e* < l or |*X*| >1}**:** inNode{ Left-daughter iTree, Right-daughter iTree, Normal, Intercept }
    - *e*← *e*+1, depth level in the decision tree
- {if *e* ≥ l or |*X*| ≤1}: exNode{Size ←|*X*|}, **end**

▼ 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

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

▼ Parameters

- *x* = one data in the tree, *x* in *X'*
- *e* = path length, set to 0 at first
- *T* = *iTree(X',e,l)* from **Algo 2**
    - T = { *exNode*{ pop |*X*/ in the node}, *InNode*{*Normal*, *Intercept*, *left*, *right* } }

▼ PathLength algorithm

- eqn (2): *H(i)*, harmonic number = ln*(i)* *+0.5772156649 ?*

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

- **While** {T = inNode} **:** PathLength(x, T.left or T.right, e ← e+1)
- {if T=exNode}: h(x)=*e*+c(T.size) **end**

Anomoly score s(n)

$$s(x,n) = 2^{-E[h(x)]/c(n)},\ E[h(x)] \approx \log(n)$$

- ▼ Paper3 MORF
  - ▼ Goal:
    - In structured data lying on manifold (easily searchable i.e. spreadsheets, image, text, NN), the indices is as important as magnitude
    - build distribution in manifold
  - ▼ Background:
    1. Classification
       - data *D_n ~ F_XY* distribution
    2. Random Forest
       - D_n ← {S^L_theta*, S^R_theta*}
       - theta* =(e*_j=feature basis, tau*= split): tend to do **halves split**
       - I(S) = Gini impurity = 1-p_i^2
    3. SPORF
  - ▼ Algo 1: Manifold Decision Tree

    function T = GROWTREE(X,y,fA,Θ)
  - ▼ Algo 2: best node split

    function (j∗,τ∗) = FINDBESTSPLIT(X,y)

    best split = minimizing Gini impurity / entropy
- ▼ Paper4 USPORF (URerF)
  - ▼ Goal:
    - estimate Geodesic distance btw 2 points
  - ▼ Parameters:

$$x = \{x_1, .., x_N\} \in \mathbb{R}^p$$

- each tree, *t*={1,..,T} has subset of sample X', |X'|= *m*<*N*
- use subset of feature *d*<*p*,
- transform X'∈ R^P into X~ ∈R^d

$$\tilde{X}_{d\times 1} = A^T X'_{p\times 1}, \quad A_{d\times p}$$

▼ Spliting Criteria

- 2-GMM Splitting with Fast-BIC

  = split with global maximum log likelihood, *s* = cluster number

▼ Algo 1: build an unsupervised random decision tree

- function t = BUILDTREE(X' subset,d allowed dimension,Θ split criteria)
- if satisfy Θ: split X' into {X'l,X'r} #note: optimal split ~ middle
- if not satisfy Θ: LeafNode(X)