

Anomaly detection with isolation forests

Which elements are few and different?

Terence Parr
University of San Francisco

Project link:
<https://github.com/parrt/msds689/blob/master/projects/iforest/iforest.md>



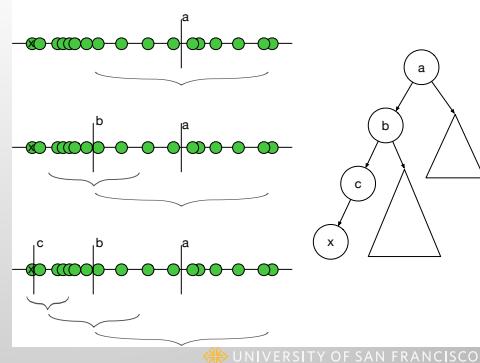
Anomaly detection

- Applications include detecting network attacks, financial fraud, unusual changes in stock prices or other timeseries / signals
- Either focus on what's normal and look for "nonnormal"
e.g., use kernel density estimates and look for elements $> 4 \sigma$
- Or, focus on the unusual, which is what isolation forests do



Random-split isolation in 1D

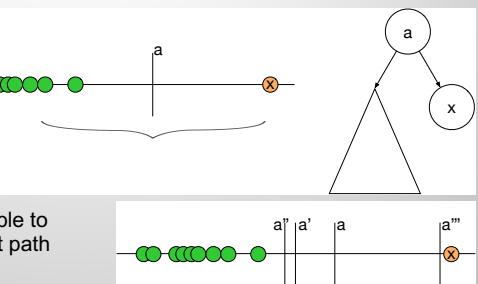
- To isolate x takes many partitions (splits at a, b, c)
- This leads to path length 3



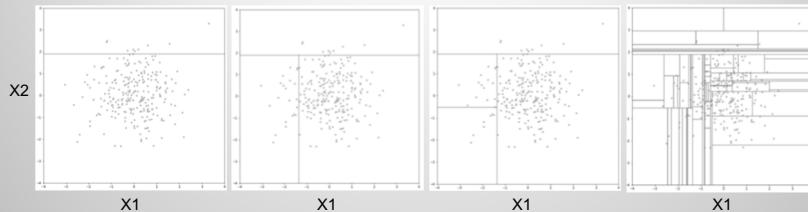
Random split on feature w/high kurtosis

x has path length 1

"Anomalies are more susceptible to isolation and hence have short path lengths." – Liu et al

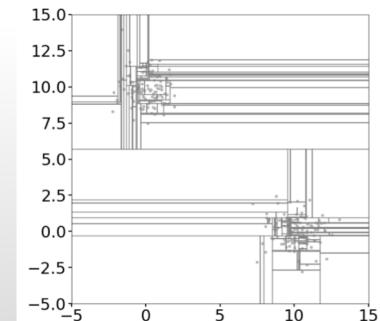


Random split isolation in 2D: [X1 X2]



Images: http://www.ncsa.illinois.edu/Conferences/LSST18/assets/pdfs/hariri_forest.pdf UNIVERSITY OF SAN FRANCISCO

Works for multimodal data too

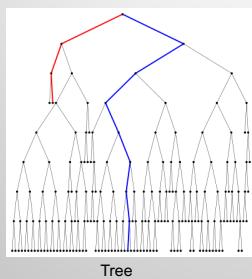


(b) Multiple Blobs

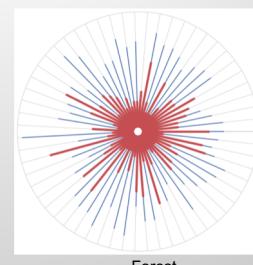
UNIVERSITY OF SAN FRANCISCO

Forest of isolation trees

- Score for x is inversely related to depth of leaf containing x
- Average score across forest to get anomaly score

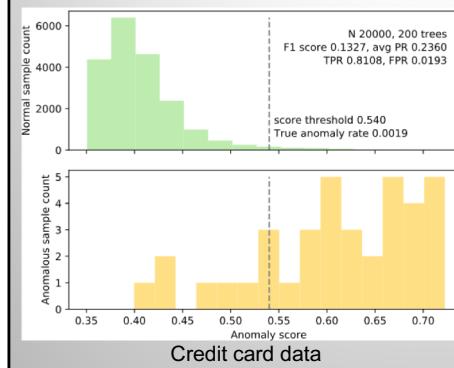


Images from Hariri et al: <https://arxiv.org/abs/1811.02141>



UNIVERSITY OF SAN FRANCISCO

Scoring: sigmoid on normalized path length



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

$$c(\psi) = \begin{cases} 2H(\psi - 1) - 2(\psi - 1)/n & \text{for } \psi > 2, \\ 1 & \text{for } \psi = 2, \\ 0 & \text{otherwise.} \end{cases}$$

$$h(i) \sim \ln(i) + 0.5772156649$$

UNIVERSITY OF SAN FRANCISCO

Tree construction

- From tiny subsample of all X training data, partition based upon random X_i feature and random split point between $\min(X_i)$ and $\max(X_i)$

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

Inputs: 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:   let  $Q$  be a list of attributes in  $X$ 
5:   randomly select an attribute  $q \in Q$ 
6:   randomly select a split point  $p$  from  $\max$  and  $\min$  values of attribute  $q$  in  $X$ 
7:    $X_l \leftarrow filter(X, q < p)$ 
8:    $X_r \leftarrow filter(X, q \geq p)$ 
9:   return  $inNode\{Left \leftarrow iTree(X_l, e + 1, l),$ 
10:    Right \leftarrow iTree(X_r, e + 1, l),
11:    SplitAtt \leftarrow q,
12:    SplitValue \leftarrow p\}
13: end if

```



UNIVERSITY OF SAN FRANCISCO

Computing path length

- Recursively count levels, e , to node containing x
- If $|X|>1$ for leaf, we need to estimate depth using $c(|X|)$ for samples X in leaf

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

Inputs : 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  $e + c(T.size)$  { $c(\cdot)$  is defined in Equation [1]}
3: end if
4:  $a \leftarrow T.splitAtt$ 
5: if  $x_a < T.splitValue$  then
6:   return  $PathLength(x, T.left, e + 1)$ 
7: else { $x_a \geq T.splitValue$ }
8:   return  $PathLength(x, T.right, e + 1)$ 
9: end if

```



UNIVERSITY OF SAN FRANCISCO